

A2

1/9/1
DIALOG(R) File 351:Derwent WPI
(c) 2000 Derwent Info Ltd. All rts. reserv.

011170213 **Image available**
WPI Acc No: 1997-148138/199714
XRPX Acc No: N97-122560

User-level process scheduler for computer system - has scheduler that performs scheduling of group of other user processes, which have priorities lower than first priority, and causes such user processes to operate or run

Patent Assignee: FUJITSU LTD (FUJIT)

Inventor: KAMADA J; ONO E; YUHARA M

Number of Countries: 003 Number of Patents: 004

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
GB 2304211	A	19970312	GB 967228	A	19960404	199714 B
JP 9054699	A	19970225	JP 95205424	A	19950811	199718
GB 2304211	B	20000419	GB 967228	A	19960404	200022
US 6108683	A	20000822	US 96621181	A	19960321	200042

Priority Applications (No Type Date): JP 95205424 A 19950811

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
GB 2304211	A	106	G06F-009/46	
JP 9054699	A	29	G06F-009/46	
GB 2304211	B		G06F-009/46	
US 6108683	A		G06F-009/46	

Abstract (Basic): GB 2304211 A

The system includes a scheduler that belongs to a user process and allocates a central processing unit (CPU) to executable ones of the processes in descending order of their priorities. The scheduler is provided in a fixed-priority process scheduler space, namely in a real-time class process scheduler space, and is given the top level of priority that would generally be assigned to a class of programs requiring real time operation. The scheduler performs the scheduling of a group of other user processes, which have priorities lower than the first priority, and causes such user processes to operate or run.

ADVANTAGE - Ensures utilisation of CPU in user process required by multi-media system while scheduling is carried by user program.

Dwg.6/39

Title Terms: USER; LEVEL; PROCESS; COMPUTER; SYSTEM; PERFORMANCE; SCHEDULE;
GROUP; USER; PROCESS; PRIORITY; LOWER; FIRST; PRIORITY; CAUSE; USER;
PROCESS; OPERATE; RUN

Derwent Class: T01

International Patent Class (Main): G06F-009/46

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C; T01-F05G5

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-54699

(43) 公開日 平成9年(1997)2月25日

(51) Int.Cl.⁶

G 0 6 F 9/46

識別記号

3 4 0

庁内整理番号

F I

G 0 6 F 9/46

技術表示箇所

3 4 0 B

審査請求 未請求 請求項の数31 O L (全 29 頁)

(21) 出願番号 特願平7-205424

(22) 出願日 平成7年(1995)8月11日

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72) 発明者 蒲田 順

神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(72) 発明者 湯原 雅信

神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(72) 発明者 小野 越夫

神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

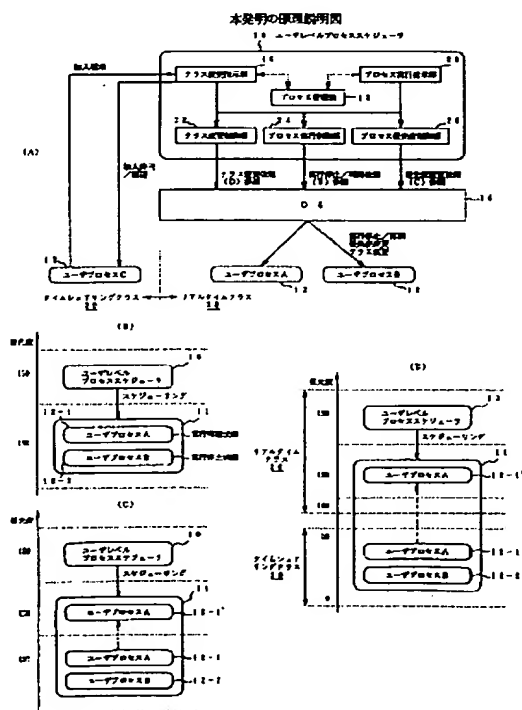
(74) 代理人 弁理士 竹内 進 (外1名)

(54) 【発明の名称】 計算機のプロセススケジューラ

(57) 【要約】

【課題】 OSがサポートするスケジューラを利用してユーザプログラムでスケジューリングを行いながらも、マルチメディアシステムが必要としているユーザプロセスでのCPU利用を保証する。

【解決手段】 OS 14によりサポートされ、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定めると共に、優先度はユーザプロセスからの指定で変更可能であり、実行可能なプロセスのうち優先度の高いプロセスからCPUを割当てて動作させる計算機のプロセススケジューラを利用する。この固定優先度プロセススケジューラ、即ちリアルタイムクラス28のプロセススケジューラ空間に、ユーザレベルプロセススケジューラ10を設ける。ユーザレベルプロセススケジューラ10は、リアルタイムクラス28の第1優先度159を持ち、第1優先度159より低い優先度を持つ他のユーザプロセス群11をスケジューリングして動作させる。即ち、CPU割当てを決めてユーザプロセスの動作をOS 14に依頼する。



1

【特許請求の範囲】

【請求項1】オペレーティングシステムによりサポートされ、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定めると共に、該優先度はユーザプロセスからの指定で変更可能であり、実行可能な複数のプロセスのうち優先度の高いプロセスからCPUを割り当てて動作させる計算機のプロセススケジューラに於いて、

ユーザプロセスに属すると共に前記固定優先度スケジューリングの第1優先度を持ち、前記第1優先度より低い優先度を持つ他のユーザプロセス群をスケジューリングして動作させる（CPU割当てを決めてユーザプロセスの動作をOSに依頼する）ユーザレベルプロセススケジューラを設けたことを特徴とする計算機のプロセススケジューラ。

【請求項2】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、第1優先度をもつ複数のユーザプロセス毎にリンクされ、各ユーザレベルプロセススケジューラは自己がリンクしたユーザプロセスをスケジューリングして動作させることを特徴とする計算機のプロセススケジューラ。

【請求項3】請求項1又は2記載の計算機のプロセススケジューラに於いて、前記第1優先度のユーザレベルプロセススケジューラは、自己がスケジュールしているユーザプロセス群中に、第2優先度以下のユーザレベルプロセススケジューラを設け、前記上位から下位に階層的に他のユーザプロセス群をスケジューリングして動作させることを特徴とする計算機のプロセススケジューラ。

【請求項4】請求項1記載の計算機のプロセススケジューラに於いて、第一優先度をもつユーザレベルプロセススケジューラを複数設け、各ユーザレベルプロセススケジューラが個別に自己のユーザプロセス群をスケジューリングして実行することを特徴とする計算機のプロセススケジューラ。

【請求項5】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスからの要求に基づくクラス変更指示部からの指示、又はプロセス管理表の参照によるプロセス実行指示部からの指示に従い、指定されたユーザプロセスの実行、停止又は再開をオペレーティングシステムに依頼するプロセス実行制御部を備えたことを特徴とする計算機のプロセススケジューラ。

【請求項6】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスからの要求に基づくクラス変更指示部からの指示、又はプロセス管理表の参照によるプロセス実行指示部からの指示に従い、指定されたユーザプロセスの優先度を第1優先度に変更してユーザプロセスを実行させ、下位の優先度に変更した他のユーザプロセスを停止させることで、ユーザプロセスの実行、停止、又

2

は再開をオペレーティングシステムに依頼するプロセス優先度制御部を備えたことを特徴とする計算機のプロセススケジューラ。

【請求項7】請求項1記載の計算機のプロセススケジューラに於いて、

前記オペレーティングシステムは、高い優先度の領域にある固定優先度のリアルタイムクラスに対し、低い優先度の領域にある優先度をプロセス状態に応じて動的に変化させるタイムシェアリングクラスを有する場合、前記ユーザレベルプロセススケジューラは、ユーザプロセスからの要求に基づくクラス変更指示部からの指示、又はプロセス管理表の参照によるプロセス実行指示部からの指示に従い、指定されたクラスの変更をオペレーティングシステムに依頼して、ユーザプロセスの実行、停止又は再開させるクラス変更制御部を備えたことを特徴とする計算機のプロセススケジューラ。

【請求項8】請求項5乃至7記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスのCPU使用時間の指定を伴う加入要求に対し、前記プロセス管理表を参照して現在加入状態にあるユーザプロセス群の要求CPU時間の合計時間を求め、該合計時間が規定時間以下であれば前記ユーザプロセスをの加入を許可し、規定値を超えていれば加入を拒否することを特徴とする計算機のプロセススケジューラ。

【請求項9】請求項8記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの加入を許可する場合、前記プロセス実行制御部に加入要求プロセスの実行停止状態への変更を指示し、且つ前記クラス変更制御部に、低い優先度の領域にある動的優先度のタイムシェアリングクラスから高い優先度の領域にある固定優先度のリアルタイムクラスへの変更を指示し、前記プロセス優先度制御部に第2優先度への変更を指示し、更にユーザプロセスをプロセス管理表に登録した後に、加入許可を通知することを特徴とする計算機のプロセススケジューラ。

【請求項10】請求項8記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの加入を許可する場合、前記クラス変更制御部にリアルタイムクラスへの変更を指示し、且つ前記プロセス優先度制御部にリアルタイムクラスの第1優先度より下位の優先度への変更を指示し、更にユーザプロセスをプロセス管理表に登録した後に、加入許可を通知することを特徴とする計算機のプロセススケジューラ。

【請求項11】請求項8記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの加入を許可する場合、前記クラス変更制御部にタイムシェアリングクラスへの変更を指示し、更にユーザプロセスをプロセス管理表に登録した後に、加入許可を通知することを特徴とする計算機のプロセススケジューラ。

【請求項12】請求項5乃至7記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスの脱会要求に対し、前記プロセス管理表からユーザプロセスを削除すると共に、前記クラス変更制御部に脱会要求を行ったユーザプロセスのタイムシェアリングクラスへの変更を指示することを特徴とする計算機のプロセススケジューラ。

【請求項13】請求項5乃至7記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、ユーザプロセスからの通知により新規なユーザプロセスの加入と脱会を処理することを特徴とする計算機のプロセススケジューラ。

【請求項14】請求項5乃至7記載の計算機のプロセススケジューラに於いて、前記クラス変更指示部は、オペレーティングシステムからの通知により新規なユーザプロセスの加入と脱会を処理することを特徴とする計算機のプロセススケジューラ。

【請求項15】請求項5乃至7記載の計算機のプロセススケジューラに於いて、前記プロセス実行制御部、クラス変更制御部、プロセス優先度制御部に対する変更要求の情報を、アプリケーションにリンクされたライブラリを経由して、アプリケーションから要求指示を受け、オペレーティングシステムに変更を依頼することを特徴とする計算機のプロセススケジューラ。

【請求項16】請求項5乃至7記載の計算機のプロセススケジューラに於いて、前記プロセス実行制御部、クラス変更制御部、プロセス優先度制御部に対する変更要求のスケジューリング仕様をユーザプロセスの1つで生成し、該ユーザプロセスのスケジューリング仕様に基づき前記ユーザレベルプロセススケジューラに変更要求を通知して前記オペレーティングシステムに変更を依頼することを特徴とする計算機のプロセススケジューラ。

【請求項17】請求項1記載の計算機のプロセススケジューラにおいて、前記ユーザレベルプロセススケジューラのスケジューリング対象となるユーザプロセスは、複数のスレッドから構成され、該ユーザプロセスの一部であるスレッドスケジューラにより前記複数のスレッドをスケジューリングすることを特徴とする計算機のプロセススケジューラ。

【請求項18】請求項1記載の計算機のプロセススケジューラに於いて、ユーザプロセスの前記ユーザレベルプロセススケジューラは、プロセス管理表に登録された1又は複数のユーザプロセスにCPU時間を割り当てる際、CPU割当時間帯にユーザプロセスが1つしか存在しないように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステムに依頼することを特徴とする計算機のプロセススケジューラ。

【請求項19】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、プロセス管理表に登録された1又は複数のユーザ

プロセスにCPU時間を割り当てる際、CPU割当時間帯の中に複数のユーザプロセスが時分割で存在するように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステムに依頼することを特徴とする計算機のプロセススケジューラ。

【請求項20】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定期間内に複数のユーザプロセスと自分自身を含めてCPU時間を順番に割り当てることにより、複数のユーザプロセスの動作を一定期間内で保証したことを特徴とする計算機のプロセススケジューラ。

【請求項21】請求項1記載の計算機のプロセススケジューラに於いて、前記プロセススケジューラは、一定の周期を設定し、各周期内で複数のユーザプロセスと自分自身を含めたCPU時間を順番に割り当てることを繰り返すことにより、複数のユーザプロセスで動作を保証したことを特徴とする計算機のプロセススケジューラ。

【請求項22】請求項21記載の計算機のプロセススケジューラに於いて、複数のユーザプロセスが持つ動作周期が異なる場合、前記ユーザレベルプロセススケジューラは、異なる動作周期の組合わせで1つの周期を決定し、該周期内で複数のユーザプロセスに使用可能なCPU時間を順番に割り当てることにより、複数のユーザプロセスの動作を保証したことを特徴とする計算機のプロセススケジューラ。

【請求項23】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定時間又は一定周期において複数のユーザプロセスに割り当てたCPU時間の占める割合が規定値以内の時、他のユーザプロセスに対するCPU時間の割当てを許容し、前記規定値を越えた時、他のユーザプロセスに対するCPU時間の割当てを拒否することを特徴とする計算機のプロセススケジューラ。

【請求項24】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定期間又は一定周期でユーザプロセスにCPU時間を割り当てる場合、連続したCPU時間を割り当てることを特徴とする計算機のプロセススケジューラ。

【請求項25】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、一定期間内又は一定周期内で複数のユーザプロセスのCPU時間を割り当てる場合、複数のユーザプロセスのCPU時間の各々を分割して割り当てることを特徴とする計算機のプロセススケジューラ。

【請求項26】請求項1記載の計算機のプロセススケジューラに於いて、プロセス管理表に登録されたユーザプロセスとして、高い優先度の領域にある固定優先度のリアルタイムクラスに属するユーザプロセスと、低い優先度の領域にあり優先度を動的に変化させるタイムシェアリングクラスに属するユーザプロセスが存在する場合、

前記ユーザレベルプロセススケジューラは、前記リアルタイムクラスのユーザプロセスに対するCPU時間の割り当てを制限してタイムシェアリングクラスのユーザプロセスにCPU時間を割り当てることを特徴とする計算機のプロセススケジューラ。

【請求項27】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスの実行停止を検出して他のユーザプロセスを動作可能とすることを特徴とする計算機のプロセススケジューラ。

【請求項28】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスにCPU時間を割り当てる際に、該ユーザプロセスに加え同じ優先度で実行可能なブロック検出プロセスを設け、前記ユーザプロセスがブロックした際に前記ブロック検出プロセスにCPU時間を割り当ててブロックを検出して通知させ、該通知により前記ユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とすることを特徴とする計算機のプロセススケジューラ。

【請求項29】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ユーザプロセスにCPU時間を割り当てる際に、該ユーザプロセスに加え低い優先度で実行可能なブロック検出プロセスを設け、前記ユーザプロセスがブロックした際に前記ブロック検出プロセスにCPU時間を割り当ててブロックを検出して通知させ、該通知により前記ユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とすることを特徴とする計算機のプロセススケジューラ。

【請求項30】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、ブロック中のユーザプロセスが実行可能な状態になったことを検出して動作させることを特徴とする計算機のプロセススケジューラ。

【請求項31】請求項1記載の計算機のプロセススケジューラに於いて、前記ユーザレベルプロセススケジューラは、オペレーティングシステムによるユーザプロセスのブロックまたはブロック回復による実行再開の通知を検出して他のユーザプロセスの実行又は停止を行うことを特徴とする計算機のプロセススケジューラ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、オペレーティングシステムでサポートされたプロセススケジューラを使用してユーザレベルのプロセススケジューラを機能させる計算機のプロセススケジューラに関し、特に、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定め、実行可能なプロセスのうち優先度の高いプロセスからCPUを割り当てて動作させるプロセススケジ

ューラを使用してユーザレベルプロセススケジューラを機能させる計算機のプロセススケジューラに関する。

【0002】

【従来の技術】計算機では、一般に複数のプログラムを平行して動作させるため、プロセスと呼ばれる実行単位が同時に複数存在できるようにしている。この時、アルゴリズムにしたがって各プロセスにCPUを割り当てる必要がある。このプロセスに対するCPUの割り当てをプロセススケジューリングという。

10 【0003】従来のプロセススケジューリングは、バッチシステムあるいはタイムシェアリングシステムであり、システム全体でのCPUの利用効率を高めることを目標として作られてきた。タイムシェアリングシステムでは、ユーザに対する応答性能を短くすることも考慮されているが、システムの負荷が高いときに応答時間が長くなることはやむなしとされ、ユーザプロセスのCPU利用に関する保証はなかった。

【0004】また、実時間性を必要とする組み込み機器の制御では、設計時に実行する全プロセスを把握することができるので、プロセスに固定した優先度をつけ、システムが優先度の高いプロセスを優先して実行する固定優先度スケジューラが用いられている。これに対し近年、マルチメディアシステムの発展により、ビデオや音声などを計算機のプロセススケジューラシステムで扱うようになってきている。ビデオや音声を滑らかに再生するためには、決まった時間間隔で一定の処理を確実に行わなければならない。そのようなマルチメディアシステムにとって、従来のスケジューリング処理は次のような問題を持っている。

30 【0005】(1)従来のバッチあるいはタイムシェアリングのスケジューリングでは、CPUがいつでもどこでも利用できるかということに関してなら保証がない。このようなスケジューリングでは、マルチメディアアプリケーションプログラムが必要とする時にCPUを利用できない。結果として、例えばビデオを再生したときにぎくしゃくした動きになってしまう。

【0006】(2)マルチメディアアプリケーションプログラムは、組み込み機器と違い、どのようなプロセスあるいはプログラムが実行されるかを予測することは出来ない。このため、予めプロセスに優先順位を設定することができない。従って、従来の固定優先度のスケジューリングでは目的を果たせない。この問題は、周期的に動作するプロセスを、周期が短いプロセスほど優先するレート・モノトニック・スケジューリングであっても同じである。

【0007】(3)マルチメディアアプリケーションプログラムは、組み込み機器と違い、プログラムの動作を信用できない。例えば、あるプログラムが故意に、または誤りから、CPUを永遠に使用してしまうと、固定優先度でスケジューリングしている場合、CPUを占有し

たプロセスより低い優先度のプロセスが全く動作しなくなってしまう。このような状況は、汎用のマルチメディアシステムでは許されない。

【0008】以上のような理由により、現在、マルチメディアシステムで利用可能な新たなプロセススケジューリングが必要となり、その研究開発が進められている。

【0009】

【発明が解決しようとする課題】マルチメディアシステムに適した新たなプロセススケジューラを導入するには、スケジューリングをオペレーティングシステム内で行う場合と、オペレーティングシステム上で動作するユーザプログラムで行う場合がある。しかし、従来のプロセススケジューラを導入するには、次のような問題があった。

【0010】(1) オペレーティングシステム内に新しいプロセススケジューラを導入する場合の問題

(a) 従来のオペレーティングシステムは、プロセススケジューラの追加や変更をオペレーティングシステム開発者以外には許していない。スケジューラを変更するには、オペレーティングシステムのソースコードや開発環境が必要である。商用オペレーティングシステムの場合、ソースコードを入手できなかったり、入手できても非常に高価なライセンス料が必要であるため、マルチメディアシステムの開発者が商用オペレーティングシステムのプロセススケジューラを変更することは事実上無理である。

【0011】(b) 非商用オペレーティングシステムでは、ソースコードを無料で入手できる場合もあるが、商用オペレーティングシステム用に作られた多数のアプリケーションプログラムを動作させることができないし、サポート体制に問題があるなど、非商用オペレーティングシステムは計算機のプロセススケジューラの非専門家を利用できる状況にない。

【0012】(c) 商用、非商用にかかわらず、オペレーティングシステムの内部を変更するには、オペレーティングシステムに関する一般的な知識と、そのオペレーティングシステムに固有の実装方法についての理解が必要であるが、そのような技術を持つものは限られている。

(2) ユーザプログラムでプロセススケジューラを導入する場合の問題

(a) ユーザプログラムやユーザプログラムが利用するライブラリで、自分のプロセス内に擬似的なプロセス(ユーザレベルスレッド)を実現し、ユーザレベルスレッドのスケジューリングを行う方法がある。この方法は、そのプロセス内のスケジューリングしか行えないため、システムに他のプロセス群が存在し、それらがオペレーティングシステムのスケジューリングのもとでCPUを利用する場合には、マルチメディアシステムが必要としているCPU利用の保証は出来ない。

【0013】(b) ユーザプログラムがスケジューラとなり、このユーザレベルスケジューラが、オペレーティングシステムでサポートする固定優先度スケジューラで使用している優先度をユーザプロセスに与えるスケジューリング方法がある。この方法は、オペレーティングシステムが提供しているスケジューリングを利用するため、マルチメディアシステムが必要としているユーザプロセスでのCPU利用の保証はできない。

(c) 更に(a)(b)では、一般にユーザプログラムの変更が必要になる。

【0014】以上のように、従来のプロセススケジューラでは、マルチメディアシステムで必要としている新しいプロセススケジューラを導入することが非常に困難である。本発明では、オペレーティングシステムがサポートするスケジューラを利用してユーザプログラムでスケジューリングを行いながらも、マルチメディアシステムが必要としているユーザプロセスでのCPU利用が保証できる計算機のプロセススケジューラを提供することを目的とする。

【0015】

【課題を解決するための手段】図1は本発明の原理説明図である。まず本発明は、オペレーティングシステム

(OS) 14によりサポートされ、スケジューリング対象となる複数のプロセスに対し優先度を固定的に定めると共に、優先度はユーザプロセスからの指定で変更可能であり、実行可能なプロセスのうち優先度の高いプロセスからCPUを割当てて動作させる計算機のプロセススケジューラを前提とする。

【0016】このような固定優先度のプロセススケジューラ、即ちリアルタイムクラス28のプロセススケジューラ空間に、本発明は、図1(A)のように、ユーザレベルプロセススケジューラ10を設けたことを特徴とする。ユーザレベルプロセススケジューラ10は、第1図(B)のように、リアルタイムクラス28の第1優先度159を持ち、第1優先度159より低い優先度を持つ他のユーザプロセス群11をスケジューリングして動作させる。即ち、CPU割当てを決めてユーザプロセスの動作をオペレーティングシステム14に依頼する。

【0017】本発明の他の形態としては、第1優先度159をもつ複数のユーザプロセス12毎にユーザレベルプロセススケジューラ10をリンクさせ、自己のユーザプロセスをスケジューリングして動作させる分散型としてもよい。また本発明の他の形態としては、第1優先度159のユーザレベルプロセススケジューラ10は、自己がスケジューリングしているユーザプロセス群中に、第2優先度158のユーザレベルプロセススケジューラを設け、上位から下位に階層的に他のユーザプロセス群をスケジューリングして動作させる階層型としてもよい。この階層法は分散法に適用し、階層分散型としてもよい。

【0018】さらに本発明の別の形態として、第一優先

度をもつユーザレベルスケジューラ10を複数設け、各ユーザレベルプロセススケジューラ10が個別に自己のユーザプロセス群11をスケジューリングして実行するようにしてもよい。ユーザレベルプロセススケジューラ10は、図1(A)のように、ユーザプロセスからの要求に基づくクラス変更指示部16からの指示、又はプロセス管理表18の参照によるプロセス実行指示部20からの指示に従い、指定されたユーザプロセスの実行、停止又は再開を、図1(B)のように、オペレーティングシステム14に依頼するプロセス実行制御部24を備える。

【0019】またユーザレベルプロセススケジューラ10は、図1(A)のように、ユーザプロセスからの要求に基づくクラス変更指示部16からの指示、又はプロセス管理表18の参照によるプロセス実行指示部20からの指示に従い、指定されたユーザプロセスAの優先度を第1優先度に変更して実行させ、他のユーザプロセスBを下位の優先度に変更して停止させるように、ユーザプロセスの実行、停止、又は再開を図1(C)のようにオペレーティングシステム14に依頼するプロセス優先度制御部26を添える。

【0020】更に、オペレーティングシステム14は、他界優先度の範囲(100~159)にある固定優先度のリアルタイムクラス28に対し、低い優先度の範囲(0~50)にあってプロセス状態に応じて動的に変化させる他のスケジューリングクラス、例えばタイムシェアリングクラス30を有する場合、ユーザレベルプロセススケジューラ10は、ユーザプロセスからの要求に基づくクラス変更指示部16からの指示、又はプロセス管理表18の参照によるプロセス実行指示部20からの指示に従い、指定されたクラスの変更をオペレーティングシステム14に依頼して、図1(D)のように、ユーザプロセスの実行、停止又は再開させるクラス変更制御部24を備える。

【0021】ここで、クラス変更指示部16は、ユーザプロセスCのCPU使用時間の指定を伴う加入要求に対し、プロセス管理表18を参照して現在加入状態にあるユーザプロセスA、Bの要求CPU時間の合計時間を求め、この合計時間が規定時間以下であればユーザプロセスCの加入を許可し、規定値を超えていれば加入を拒否する。

【0022】加入処理は、クラス変更指示部16がユーザプロセスCの加入を許可する場合、プロセス実行制御部24に加入要求プロセスCの実行停止状態への変更を指示し、且つクラス変更制御部22にリアルタイムクラスへの変更を指示し、更にプロセス優先度制御部26に第2優先度への変更を指示し、更にユーザプロセスCをプロセス管理表18に登録した後に、加入許可を通知する。

【0023】また加入処理の別の形態として、クラス変

更指示部16は、ユーザプロセスCの加入を許可する場合、クラス変更制御部22にタイムシェアリングクラス30からリアルタイムクラス28への変更を指示し、且つプロセス優先度制御部26にリアルタイムクラスの例えば最下位優先度100への変更を指示し、更にユーザプロセスCをプロセス管理表18に登録した後に、加入許可を通知する。

【0024】更に、加入処理の別の形態としてクラス変更指示部16は、ユーザプロセスCの加入を許可する場合、クラス変更制御部22にタイムシェアリングクラス30への変更を指示し、更にユーザプロセスCをプロセス管理表18に登録した後に、加入許可を通知する。ユーザプロセスの脱会要求があると、クラス変更指示部16は、プロセス管理表18からユーザプロセスを削除すると共に、クラス変更制御部22に脱会要求を行ったユーザプロセスのタイムシェアリングクラス30への変更を指示する。

【0025】クラス変更指示部16は、ユーザプロセスからの通知により新規なユーザプロセスの加入と脱会を処理できるし、またオペレーティングシステム14からの通知により新規なユーザプロセスの加入と脱会を処理することもできる。更に、ユーザレベルプロセススケジューラ10のプロセス実行制御部24、クラス変更制御部22、プロセス優先度制御部26に対する変更要求の情報をライブラリに格納し、アプリケーションからの要求指示でライブラリからユーザレベルプロセススケジューラ10に、対応する変更要求のメッセージを通知してオペレーティングシステムに変更を依頼することもできる。

【0026】またプロセス実行制御部24、クラス変更制御部22、プロセス優先度制御部26に対する変更要求のスケジューリング仕様をユーザプロセスの1つで生成し、ユーザプロセスのスケジューリング仕様に基づきユーザレベルプロセススケジューラ10に変更要求を通知して、オペレーティングシステム14に変更を依頼する。

【0027】ここで、ユーザレベルプロセススケジューラ10のスケジューリング対象となるユーザプロセス12は、複数のスレッドから構成され、ユーザプロセスの一部であるスレッドスケジューラにより複数のスレッドをスケジューリングする。ユーザプロセスのユーザレベルプロセススケジューラ10は、プロセス管理表18に登録された1又は複数のユーザプロセスにCPU時間を割り当てる際、CPU割当対象となるユーザプロセスが1つしか存在しないように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステム10に依頼する。

【0028】またユーザレベルプロセススケジューラ10は、1又は複数のユーザプロセスにCPU時間を割り当てる際、同じCPU割当時間帯に複数のユーザプロセ

スが時分割で存在するように、ユーザプロセスの実行と停止、優先度変更、又はクラス変更をオペレーティングシステム14に依頼する。ユーザレベルプロセススケジューラ10は、一定期間内に複数のユーザプロセスと自分自身を含めてCPU時間を順番に割り当てることにより、複数のユーザプロセスの動作を一定期間内で保証する。またユーザレベルプロセススケジューラ10は、一定の周期を設定し、各周期内で複数のユーザプロセスと自分自身を含めてCPU時間を順番に割り当てることを繰り返すことにより、複数のユーザプロセスで動作を保証する。

【0029】複数のユーザプロセスが持つ動作周期が異なる場合、ユーザレベルプロセススケジューラ10は、異なる動作周期の組合わせにより1つの周期を決定し、この周期内で複数のユーザプロセスに使用可能なCPU時間を順番に割り当て、複数のユーザプロセスの動作を保証する。ユーザレベルプロセススケジューラ10は、一定時間又は一定周期において複数のユーザプロセスに割り当てたCPU時間の占める割合が規定値以内の時、他のユーザプロセスに対するCPU時間の割当てを許容し、規定値を越えた時、他のユーザプロセスに対するCPU時間の割当てを拒否するアドミSSIONコントロールを行う。

【0030】ユーザレベルプロセススケジューラ10は、一定期間又は一定周期でユーザプロセスにCPU時間を割り当てる場合、連続したCPU時間を割り当ててもよいし、複数のユーザプロセスのCPU時間の各々を分割して割り当ててもよい。プロセス管理表18に登録されたユーザプロセスとして、高い優先度の範囲(100~159)にある固定優先度のリアルタイムクラス28に属するユーザプロセスと、低い優先度の範囲(0~50)にあつて動的に優先度を変化させるタイムシェアリングクラスに属するユーザプロセス30が存在する場合、ユーザレベルプロセススケジューラ10は、リアルタイムクラス28のユーザプロセスに対するCPU時間の割当てを制限して、タイムシェアリングクラス30のユーザプロセスにCPU時間を割り当てる。

【0031】ユーザレベルプロセススケジューラ10は、ユーザプロセスの実行停止を検出して他のユーザプロセスを動作可能とする。ユーザレベルプロセススケジューラ10は、ユーザプロセスにCPU時間を割り当てる際に、ユーザプロセスに加え同じ優先度で実行可能なブロック検出プロセスを設け、ユーザプロセスがI/O待ち等でブロックした際に、ブロック検出プロセスにCPU時間を割り当ててブロック発生を検出して通知させ、この通知によりユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とする。

【0032】またユーザレベルプロセススケジューラ10は、ユーザプロセスにCPU時間を割り当てる際に、更に、ユーザプロセスより低い優先度で実行可能なプロ

ック検出プロセスを設け、ユーザプロセスがI/O待ち等でブロックした際にブロック検出プロセスにCPU時間を割り当ててブロック発生を検出して通知させ、この通知によりユーザプロセスが実行停止したことを認識して他のユーザプロセスを実行可能とする。

【0033】ユーザレベルプロセススケジューラ10は、ブロック中のユーザプロセスが実行可能な状態になったことを検出して動作させる。ユーザレベルプロセススケジューラ10は、オペレーティングシステム14によるユーザプロセスのブロックまたはブロック回復による実行再開の通知を検出して他のユーザプロセスの実行又は停止を行うこともできる。

【0034】このような本発明のユーザレベルプロセススケジューラによれば、次の利点が得られる。

(1) スケジューリング対象のプロセスに、常に一定割合のCPUを割り当て、CPU利用を保証するようなスケジューラを実現できる。

(2) 従来のプログラムを変更せずにユーザプロセスをスケジューリング対象とすることができる。新しいスケジューラのためにプログラムを書き直したり、リンクし直したりする必要がない。

【0035】(3) 新たなスケジューラを実現するのにオペレーティングシステムのソースコードを必要としない。

(4) 商用のオペレーティングシステム上で利用できる。

(5) 新たなスケジューリングを実現するために、オペレーティングシステム内部の知識を必要としない。

【0036】(6) 各種のスケジューラを試す時に、システムを再起動する必要がない。

(7) 振る舞いが信用できないプログラムをスケジューリングの対象としても、他のプログラムの実行を進めることのできるスケジューラを実現できる。この結果、マルチメディア処理で必要とされる一定割合のCPU利用を保証するスケジューラなど、ユーザのニーズに合わせてさまざまなスケジューラを商用オペレーティングシステムの上に実装できる。

【0037】また、振る舞いの信用できないプロセスをスケジューリング対象とすることができるなど、オペレーティングシステム内で実現するのと同等のスケジューラをユーザレベルのスケジューラとして実装することができる。

【0038】

【発明の実施の形態】

<目次>

1. 動作環境
2. ユーザレベルプロセススケジューラの機能と動作
3. ユーザプロセスの加入脱会
4. ユーザプロセスのCPU利用の保証
5. ユーザプロセスのブロック検出

1. 動作環境

本発明のユーザレベルプロセススケジューラは、オペレーティングシステムが固定優先度のスケジューリングを提供していることを前提に実現することができる。固定優先度スケジューリングは、組込用のリアルタイムオペレーティングシステムでは以前から提供されており、また最近の汎用オペレーティングシステムとして知られているUNIX System V、Sun OS 5、Windows/NTでも提供されるようになってきている。

【0039】オペレーティングシステムの標準を規定するPOSIX（ユニックス標準の統合組織）では、固定優先度スケジューリングを行うスケジューラを規定している。サンマイクロシステム社のオペレーティングシステムSolaris 2.4は、POSIXに準拠しており、リアルタイムクラスと呼ばれる固定優先度スケジューラをもっている。このリアルタイムクラスでは、優先度として100～159の値を使用することができる。優先度は、値が大きい方が優先して実行されることを意味する。以下の実施例にあっては、特に断わらない限り、オペレーティングシステムSolaris 2.4上での動作環境を例にとるものとする。

【0040】図2は本発明のユーザレベルプロセススケジューラの動作環境の実施形態である。図2において、オペレーティングシステムで提供される固定優先度スケジューリングのリアルタイムクラスの空間において、本発明のユーザレベルプロセススケジューラ10はリアルタイムの最高優先度である優先度159で動作するように優先度が設定され、スケジューリング対象となるユーザプロセス群11のユーザプロセス12-1、12-2を第2優先度となる優先度158で動作させるように設定している。

【0041】図3は本発明によるユーザレベルプロセススケジューラの動作環境の他の実施形態であり、この実施形態にあっては、リアルタイムクラスの優先度159で動作するユーザレベルプロセススケジューラは、ユーザレベルプロセススケジューラ10-1、10-2の2つに分割されている。ユーザレベルプロセススケジューラ10-1、10-2のそれぞれは、スケジューリング対象であるユーザプロセス12-1、12-2にライブラリとしてそれぞれリンクされている。このためユーザレベルプロセススケジューラ10-1、10-2は、それぞれリンクされているユーザプロセス12-1、12-2の動作を制御する。

【0042】図4は本発明によるユーザレベルプロセススケジューラの他の実施形態であり、この実施形態にあっては、ユーザレベルプロセススケジューラが階層的にスケジューリングを行うことを特徴とする。まずユーザレベルプロセススケジューラ10-11は、リアルタイムクラスの第1優先度である優先度159で動作するよ

うに優先度が設定される。

【0043】優先度159を設定したユーザレベルプロセススケジューラ10-11のスケジューリング対象として、第2優先度となる優先度158を設定した第2のユーザレベルプロセススケジューラ10-12及びユーザプロセス12-1を設けて、それぞれ動作させている。更に、第2のユーザレベルプロセススケジューラ10-12は、スケジューリング対象として優先度157を設定したユーザプロセス12-2、12-3をもち、それぞれ優先度157で動作させる。

【0044】図5はユーザレベルプロセススケジューラによる階層的なスケジューリングの他の実施形態であり、この実施形態にあっては、図3の実施形態について階層的なスケジューリング構造としたことを特徴とする。即ち、リアルタイムクラスの優先度159で動作するユーザプロセス12-1、12-2のそれぞれにライブラリとしてリンクされたユーザレベルプロセススケジューラ10-1、10-2は、優先度158を設定して動作するユーザプロセス12-3、12-4にライブラリとしてリンクしたユーザレベルプロセススケジューラ10-3、10-4を設けている。

【0045】即ち、ユーザレベルプロセススケジューラ10-1、10-2は、第1階層となる優先度159でユーザプロセス12-1、12-2をスケジューリングし、ユーザプロセス12-1、12-2のスケジューリング方針に従って、第2階層となる優先度158のユーザレベルプロセススケジューラ10-3、10-4がユーザプロセス12-3、12-4をスケジューリングすることになる。

【0046】図6は本発明によるユーザレベルプロセススケジューラの他の実施形態であり、図2の実施形態をリアルタイムクラスについて複数系統設けたことを特徴とする。即ち、リアルタイムクラスの優先度159で動作する2つのユーザレベルプロセススケジューラ10-1、10-2を設けると共に、それぞれのスケジューリング対象となるユーザプロセス群11-1、11-2として、優先度158で動作するユーザプロセス12-1、12-2及びユーザプロセス12-3、12-4を設けている。このような実施形態にあっては、ユーザレベルプロセススケジューラ10-1がユーザプロセス12-1、12-2をスケジューリングし、またユーザレベルプロセススケジューラ10-2がユーザプロセス12-3、12-4をスケジューリングすることになる。もちろん図6はオペレーティングシステム内に2つのユーザレベルプロセススケジューラ10-1、10-2が存在する場合であるが、必要に応じて任意の数のユーザレベルプロセススケジューラを存在させることができる。

2. ユーザレベルプロセススケジューラの機能と動作
次に図2の実施形態を対象に、本発明により提供される

ユーザレベルプロセススケジューラの機能と動作を説明する。図7は、リアルタイムクラスの第1優先度となる優先度159で動作するユーザレベルプロセススケジューラ10の詳細である。ユーザレベルプロセススケジューラ10は、クラス変更指示部16、プロセス管理表18、プロセス実行指示部20、クラス変更制御部22、プロセス実行制御部24及びプロセス優先度制御部26で構成される。

【0047】クラス変更指示部16は、ユーザレベルプロセススケジューラ10が動作しているスケジュール空間であるリアルタイムクラス28以外のスケジューリングクラス、例えばタイムシェアリングクラス30で動作しているユーザプロセス12-3からの加入要求を受け、プロセス管理表18を参照して、予め定めた条件に従って許可または拒否を決定する。

【0048】許可の場合にはクラス変更制御部22にタイムシェアリングクラス30からリアルタイムクラス28へのクラス変更を指示し、ユーザプロセス12-3に許可を通知する。加入要求に対する許可の条件としては、例えばプロセス管理表18で管理しているユーザプロセス群の合計CPU時間である。ユーザプロセス12-3は加入要求と同時に自己のCPU時間を指定し、この指定時間を現在プロセス管理表18で管理しているユーザプロセスのCPU時間に加えて、規定時間以内であれば加入を許可し、超えていれば加入を拒否するようになる。

【0049】ここでプロセス管理表18は、図8に示すように、プロセス識別子と例えば動作可能状態としたユーザプロセスを一定周期内で動作させる場合、1周期内のCPU使用時間を登録している。1周期内のCPU使用時間は、例えば100%で100msとしている。この例では、図7のユーザプロセス12-1、12-2、12-3のプロセス識別子をA、B、Cとすると、ユーザプロセスC12-3が加入要求を行ったときに、既にプロセス管理表18には動作可能状態にあるユーザプロセス12-1、12-2のプロセス識別子A、Bとその1周期内のCPU使用時間50ms、20msが登録されている。

【0050】この状態でプロセス識別子Cをもつユーザプロセス12-3の加入要求があると、加入要求に伴ってCPU使用時間10msが指定されていることから、合計CPU使用時間として80msが求められ、これは全CPU使用時間の80%であることから加入要求を許可し、プロセス管理表18にプロセス識別子CとそのCPU使用時間10msを図示のように登録することになる。

【0051】再び図7を参照するに、プロセス実行指示部20はプロセス管理表18を参照し、次にどのプロセスをどれだけの時間実行するかを決定し、クラス変更制御部22に変更のための指示を行う。即ちプロセス実行

指示部20は、クラス変更制御部22に対しては、タイムシェアリングクラス30とリアルタイムクラス28との間のクラス変更の指示を行う。

【0052】またプロセス実行指示部20は、プロセス実行制御部24に対しては、実行、停止または再開の指示を行う。更にプロセス実行指示部20は、プロセス優先度制御部26に対しては、リアルタイムクラス28における優先度159~100内における変更を指示する。クラス変更制御部22は、クラス変更指示部16またはプロセス実行指示部20からのクラス変更の指示に従い、指定されたユーザプロセスのクラス変更をオペレーティングシステム14に依頼する。このクラス変更の依頼を受けて、オペレーティングシステム14は、固定優先度スケジュールクラスであるリアルタイムクラス28と、それ以外のスケジュールクラスであるタイムシェアリングクラス30との間のクラス変更を実行する。

【0053】プロセス実行制御部24は、クラス変更指示部16またはプロセス実行指示部20の指示に従い、指定されたユーザプロセスの実行、停止または再開をオペレーティングシステム14に依頼する。更にプロセス優先度制御部26は、クラス変更指示部16またはプロセス実行指示部20の指示に従い、固定優先度であるリアルタイムクラス28における優先度の変更をオペレーティングシステム14に依頼する。

【0054】ユーザレベルプロセススケジューラ10に設けたクラス変更制御部22、プロセス実行制御部24及びプロセス優先度制御部26は、そのときのクラス変更指示部16またはプロセス実行指示部20からの指示により個別に動作して、ユーザプロセスのスケジューリングの変更を行う。そこで、これら3つの制御部22、24、26のそれぞれについてユーザプロセスの変更制御を説明する。

【0055】図9は図7のプロセス実行制御部24によるユーザプロセスの状態変更の実施形態であり、オペレーティングシステム14が提供するシグナル機能を用いてユーザプロセスの状態変更を行うことで、ユーザレベルプロセススケジューラがユーザプロセスの実行制御を行うことを特徴とする。図9において、優先度159のユーザレベルプロセススケジューラ10は、スケジューリング対象としてユーザ部11にプロセス識別子A、Bをもつユーザプロセス12-1、12-2をもっている。このユーザプロセス12-1、12-2の実行可能状態への変更は、オペレーティングシステム14が提供するシグナル機能であるSIGCONTを引数としてkillシステムコールを発行することで実現できる。

【0056】またユーザプロセス12-1、12-2の実行停止状態への変更は、SIGSTOPを引数としてkillシステムコールを発行することで実現できる。例えば図9のように、ユーザレベルプロセススケジューラ10がプロセス識別子Aをもつユーザプロセス12-

17

1 を実行可能状態とし、プロセス識別子 B をもつユーザプロセス 12-2 を実行停止状態とすることで、ユーザプロセス 12-1 だけを動作させることができる。

【0057】図 10 のフローチャートは、図 7 のプロセス実行指示部 20 からの実行指示に基づくプロセス実行制御部 24 の処理動作である。プロセス実行制御部 20 は、例えばオペレーティングシステム 14 からの SIGCONT を引数とした kill システムコールを受けると、まずステップ S1 で、プロセス管理表 18 のインデックスを先頭のエン트리とする。

【0058】ステップ S2 で、インデックスの示すエントリは現在使用中（登録中）にあるか否かチェックする。使用中にあれば、ステップ S3 で、インデックスの示すプロセス識別子のユーザプロセスの実行状態への変更をプロセス実行制御部 24 に指示する。例えば図 8 のプロセス管理表 18 の場合には、最初のインデックスで指定されるユーザプロセス A の実行をプロセス実行制御部 24 に指示する。

【0059】このためプロセス実行制御部 24 は、ステップ S24 でユーザプロセス A の CPU 使用時間である 50ms 間スリープすることで、ユーザプロセス A を 50ms 間と動作させる。続いてステップ S5 で、インデックスの示すユーザプロセス A の実行停止状態への変更をプロセス実行制御部 24 に依頼し、これを受けてオペレーティングシステム 14 はユーザプロセス A の実行を停止する。ステップ S3～S4 で最初のインデックスのユーザプロセス A の動作が済むと、ステップ S6 で、インデックスが最後のエントリを指しているか否かチェックし、指していなければ、ステップ S7 でインデックスを 1 つ進め、ステップ S2 に戻り、次の 2 番目インデックスの示すエントリのユーザプロセス B の処理を同様に行う。

【0060】ステップ S6 でインデックスの最後のエントリが判別されると、再びステップ S1 に戻り、プロセス管理表 18 の先頭のインデックスからの処理を繰り返す。これにより 1 周期内でプロセス管理表 18 に登録されて動作可能状態にある複数のユーザプロセスを、設定された CPU 時間で動作させるスケジューリングを行うことができる。

【0061】図 11 は、図 7 のプロセス優先度制御部 26 によるユーザプロセスの変更の実施形態である。この実施形態にあつては、オペレーティングシステム 14 が提供する priocntl システムコールを使用し、ユーザプロセスの優先度を変更することで、ユーザレベルプロセススケジューラがユーザプロセスの実行制御を行うことを特徴とする。

【0062】例えば図 11 のように、ユーザレベルプロセススケジューラ 10 が優先度 157 にあるプロセス識別子 A のユーザプロセス 12-1 を優先度 158 に変更することで、プロセス識別子 A のユーザプロセス 12-

18

1' だけを動作させることができる。図 12 は、図 7 のプロセス実行指示部 20 からの優先度変更指示に基づくプロセス優先度制御部 26 の処理動作を示す。

【0063】まずステップ S1 で、優先度変更指示に対しプロセス管理表 18 のインデックスを先頭にセットし、ステップ S2 で、インデックスの示すエントリが使用中か否かチェックする。使用中にあれば、ステップ S3 で、インデックスで示すユーザプロセスの優先度 158 への変更をプロセス優先度制御部 26 に指示する。ステップ S4 で、インデックスが示すユーザプロセスの CPU 時間をスリープすることで、ユーザプロセス C を指定時間動作させる。更にステップ S5 で、インデックスの示すユーザプロセスの優先度 157 への変更をプロセス優先度制御部 26 に指示する。なおステップ S5 にあつては、停止動作を確実にするため、リアルタイムクラスの最下位の優先度である優先度 100 への変更をプロセス優先度制御部 26 に指示するようにしてもよい。

【0064】続いてステップ S6 で、インデックスが最後のエントリを指しているか否かチェックし、指していなければ、ステップ S7 でインデックスを 1 つ進め、再びステップ S2 に戻り、プロセス管理表の次のインデックスのエントリに登録されているユーザプロセスについて、ステップ S3～S5 の優先度の変更による実行と停止を繰り返す。

【0065】図 13 は、図 7 のプロセス実行指示部 20 からのクラス変更指示に基づくクラス変更制御部 22 のユーザプロセスの変更処理の実施形態である。本発明が実現されるオペレーティングシステム Solaris 2.4 は、固定優先度のリアルタイムクラス 28 に加えてタイムシェアリングスケジューリングを行うタイムシェアリングクラス 30 をもっている。タイムシェアリングクラス 30 にあつては、オペレーティングシステム 14 が次のようにユーザプロセスの優先度を動的に変更する。ここでタイムシェアリングクラス 30 は、優先度 59～0 をもつ。

【0066】（1）ユーザプロセスが一定時間 CPU を使用した場合、その優先度を一定値下げる。

（2）ユーザプロセスが実行可能であるにも関わらず一定時間 CPU 割当てが行われなかった場合、その優先度を一定値上げる。

（3）ユーザプロセスが自発的に CPU 割当てを手放した場合、即ちスリープした場合、ウェイクアップ後にその優先度を一定値上げる。

【0067】このようなタイムシェアリングクラス 30 と本発明のユーザレベルプロセススケジューラが動作しているリアルタイムクラス 28 のクラス変更として、オペレーティングシステム 14 が提供する priocntl システムコールを使用してクラス変更を行うことで、ユーザレベルプロセススケジューラがユーザプロセスの実行制御を行うことができる。

【0068】例えば図13のように、ユーザレベルプロセススケジューラ10がタイムシェアリングクラスで動作していたプロセス識別子Aのユーザプロセス12-1をリアルタイムクラス28の優先度158に変更することで、プロセス識別子Aのユーザプロセス12-1のみを動作させることができる。図14のフローチャートは、図7のプロセス実行指示部20よりクラス変更指示をクラス変更制御部22に対し行ったときのユーザプロセスの変更処理である。

【0069】オペレーティングシステムが提供するpriorityシステムコールを使用してプロセス実行指示部20に対しクラス変更指示が行われると、まずステップS1でプロセス管理表18のインデックスを先頭にセットし、ステップS2で、インデックスの示すエントリが使用中か否かチェックする。使用中にあれば、ステップS3で、インデックスの示すユーザプロセスのリアルタイムクラス28への変更をクラス変更指示部22に指示する。

【0070】これによりオペレーティングシステム14は、対象となったユーザプロセスをリアルタイムクラスの優先度158に変更し、ステップS4で、インデックスが示すユーザプロセスのCPU時間をスリープすることで、ユーザプロセスを指定時間動作させる。ステップS5で、インデックスの示すユーザプロセスのタイムシェアリングクラス28への変更をクラス変更制御部22に指示し、停止状態とする。

【0071】続いてステップS6で、インデックスが最後のエントリを指しているか否かチェックし、指していなければ、ステップS7でインデックスを1つ進め、ステップS2に戻って次のユーザプロセスについて同様な処理を繰り返す。

3. ユーザプロセスの加入脱会

図7のリアルタイムクラス28のスケジュール空間にあるユーザレベルプロセススケジューラ10のスケジュール対象となるユーザプロセスへの加入は、タイムシェアリングクラス30に存在するユーザプロセスからの加入要求を受けて処理するようになる。またリアルタイムクラス28でスケジューリングされているユーザプロセスの脱会は、ユーザプロセスからの脱会要求を受けて処理するようになる。

【0072】このユーザプロセスの加入要求及び脱会要求の処理はクラス変更指示部16に対し行われ、クラス変更指示部16は、クラス変更制御部22、プロセス実行制御部24またはプロセス優先度制御部26のいずれかに対する指示で3つの異なった加入脱会処理を行うことができる。図15のフローチャートは、プロセス実行制御部24に対する実行、停止または再開の指示を利用したユーザプロセスの加入脱会処理である。まずステップS1で、タイムシェアリングクラス30にあるユーザプロセスからの加入または脱会要求を待っている。ステ

ップS2で、要求があると加入要求か否かチェックする。

【0073】いまタイムシェアリングクラス30にあるプロセス識別子Cのユーザプロセス12-3から加入要求が行われたとする。この実施形態にあっては、ユーザプロセス12-3は加入要求に際し自分自身の1周期内のCPU使用時間を例えば10msとして指定して加入を要求している。加入要求を受けたことでステップS3に進み、クラス変更指示部16はプロセス管理表18に登録されているユーザプロセスのCPU使用時間の合計と加入要求プロセスの要求CPU時間の和が、予め定めた所定時間100msより小さいか否かチェックする。

【0074】ここで図8のプロセス管理表18のように、プロセス識別子A、Bのユーザプロセス12-1、12-2が登録されており、それぞれの1周期内のCPU時間が50ms、20msであったとすると、加入要求を行ったユーザプロセス12-3のCPU使用時間10msを加えた合計時間は80msとなり、100msより小さいことから加入を許可するため、ステップS4に進む。

【0075】ステップS4では、加入要求を行ったユーザプロセスの実行停止状態への変更をプロセス実行制御部24に指示し、オペレーティングシステム14への依頼で、加入要求を行ったユーザプロセス12-3を停止状態とする。続いて、加入要求を行ったユーザプロセス12-3のリアルタイムクラス28への変更をステップS5でクラス変更制御部22に指示する。

【0076】更にステップS6で、加入要求を行ったユーザプロセス12-3への優先度158への変更をプロセス優先度制御部26に指示する。これにより加入要求を許可されたユーザプロセス12-3は、ユーザレベルプロセススケジューラ10が動作しているリアルタイムクラス28の優先度158に設定される。続いてステップS7で、加入要求を行ったユーザプロセス12-3を図8のプロセス識別子CとそのCPU時間10msに示すようにプロセス管理表18に登録する。

【0077】最終的にステップS5で、加入要求を行ったユーザプロセス12-3に対し加入許可を通知する。一方、ステップS3で加入要求を行ったユーザプロセスを含めた合計CPU時間が100msを超えていた場合には、ステップS9に進み、加入要求を行ったユーザプロセス12-3に対し加入拒否を通知する。またユーザプロセスからの脱会要求があった場合には、ステップS2からS10に進み、脱会要求を行ったユーザプロセスをプロセス管理表18から削除し、ステップS11で、脱会要求をしたユーザプロセスのタイムシェアリングクラス30への変更をクラス変更制御部22に指示する。

【0078】図16のフローチャートは、ユーザプロセスからの加入要求に対しプロセス優先度制御部26による優先度の変更制御を利用して加入させる場合の処理動

21

作である。まずステップS 1でユーザプロセスからの加入または脱会要求を待ち、要求があるとステップS 2で加入要求か否か判別し、加入要求であれば、ステップS 3で、プロセス管理表18の登録済みユーザプロセスのCPU使用時間と加入要求プロセスの要求CPU時間との和が100msより小さいか否かチェックし、小さければ加入許可のため、ステップS 4に進む。

【0079】ステップS 4では、加入要求プロセスのリアルタイムクラス28への変更をクラス変更制御部22に指示し、ステップS 5で、加入要求プロセスの優先度100への変更をプロセス優先度制御部26に指示し、ステップS 6で、加入要求をしたユーザプロセスをプロセス管理表18に登録した後、ステップS 7で加入許可を通知する。ステップS 3で100ms以上であった場合は、ステップS 7で加入拒否を通知する。

【0080】またステップS 2で脱会要求であった場合には、ステップS 9でプロセス管理表18からの削除と、ステップS 10のタイムシェアリングクラス30へのクラス変更の指示を行う。図17のフローチャートは、ユーザプロセスからの加入または脱会要求に対し図7のクラス変更制御部22の制御機能を利用して行う処理動作である。ステップS 1でユーザプロセスからの加入または脱会要求を待ち、要求があると、ステップS 2で加入要求か否か判別し、加入要求であればステップS 3に進み、プロセス管理表18のCPU使用時間と加入要求プロセスの要求CPU時間の合計が100msより小さいか否かチェックする。

【0081】合計時間が100msより小さければ加入許可のため、ステップS 4に進み、加入要求を行ったユーザプロセスのタイムシェアリングクラス30への変更をクラス変更制御部22に指示する。続いてステップS 5で、加入要求を行ったユーザプロセスをプロセス管理表18に登録し、ステップS 6で加入許可を通知する。この場合には、加入要求を行ったユーザプロセスは最初タイムシェアリングクラス30に置かれて、実行の際にユーザレベルプロセススケジューラ10によるリアルタイムクラス28にクラス変更されることになる。

【0082】一方、ステップS 3で100ms以上であった場合には、ステップS 7で加入拒否を通知する。また脱会要求であった場合には、ステップS 8でプロセス管理表18から削除した後、ステップS 9で、タイムシェアリングクラス30へのクラス変更を指示する。図18は、ライブラリによるユーザレベルプロセススケジューラに対するスケジューリング及び優先度変更要求の通知の実施形態である。

【0083】図18において、アプリケーションプログラム32にリンクされるライブラリ34にユーザレベルプロセススケジューラ10のスケジューリング方式、即ちリアルタイムクラスかタイムシェアリングクラスかの指定情報、及びまたは優先度を変更する関数を用意す

22

る。このライブラリ34の内容が呼び出された場合、メッセージによりユーザレベルプロセススケジューラ10に変更を通知する。メッセージによる変更通知を受けたユーザレベルプロセススケジューラ10は、変更が可能かどうかを判断し、可能ならばオペレーティングシステムから提供されるpriorityシステムコールにより変更する。変更が不可能ならばその旨をライブラリ34に通知し、更にライブラリ34がアプリケーションプログラム32に通知する。

10 【0084】図19は、ユーザレベルプロセススケジューラに対するライブラリによるスケジューリング方式及びまたは優先度変更要求の通知の他の実施形態である。この図19の実施形態にあっては、ユーザプロセス12を生成する親プロセス36が、リンクするライブラリ38を通してユーザレベルプロセススケジューラ10に、予め定めたスケジューリング仕様に従ったメッセージを伝えてスケジューリング方式及び優先度変更要求を行うようにしている。

20 【0085】親プロセス36で作成されるスケジューリング仕様40は、例えば図20に示すように、ユーザプロセス12が動作し始めてからの相対的な開始時刻、終了時刻、及びその間のスケジューリング方式を記述している。このようなスケジューリング仕様40に従ったライブラリ38からのメッセージを受けたユーザレベルプロセススケジューラ10は、スケジューリング仕様40に従ったユーザプロセス12のスケジューリングを実行することになる。

30 【0086】図21は、本発明のユーザレベルプロセススケジューラでスケジューリングされるユーザプロセスの内部構造を示している。本発明が対象とするユーザレベルプロセススケジューラのスケジューリング対象となるユーザプロセス12は、図21のようにマルチスレッドライブラリ42をリンクしており、スレッド群44の中に例えばスレッド46-1、46-2、46-3の3つのスレッドがマルチスレッドライブラリ42によりスケジューリングされている。

4. ユーザプロセスのCPU利用の保証

図7に示す制御機能をもつユーザレベルプロセススケジューラ10は、スケジューリング対象となっているユーザプロセス即ちプロセス管理表18に登録されているユーザプロセスを対象に、一定割合のCPU割当てを行ってCPU利用を保証するようなスケジューリングを実行できる。

50 【0087】図22は、CPU割当て対象となるユーザプロセスを1つとした場合の動作形態の説明図である。図22にあっては、ユーザレベルプロセススケジューラ10が動作したいプロセス識別子Aのユーザプロセス12-1を実行可能状態に変更する指示を行う。この変更指示はプロセス実行指示部20により行われ、プロセス実行制御部24に対する実行/停止指示、クラス変更制

御部 22 に対するクラス変更指示、またはプロセス優先度制御部 26 に対する優先度変更指示のいずれかとなる。

【0088】プロセス実行指示部 20 によるプロセス実行制御部 24 に対する指示については、kill システムコールによりユーザプロセス 12-1 だけを実行状態とし、その他のユーザプロセスを実行停止状態とする。またプロセス実行指示部 20 からのプロセス優先度制御部 26 に対する優先度変更指示については、prioctl システムコールによりユーザプロセス 12-1 だけを優先度 158 に設定し、その他のユーザプロセスを優先度 157 に設定する。

【0089】更に、プロセス実行指示部 20 によるクラス変更制御部 22 に対するクラス変更指示については、prioctl システムコールによりユーザプロセス 12-1 だけをリアルタイムクラス 28 に変更し、その他のユーザプロセスをタイムシェアリングクラス 30 に変更する。以上のユーザプロセスの変更指示が済んだならば、ユーザレベルプロセススケジューラ 10 はユーザプロセス 12-1 を動作させたい時間だけ、即ちプロセス管理表 18 の参照で得られた CPU 割当時間だけ usleep システムコールによりスリープすることで、図 22 のようにユーザプロセス 12-1 を動作させる。次のプロセス識別子 B をもつユーザプロセス 12-2 についても同様であり、プロセス管理表 18 に登録されて動作可能状態にある複数のユーザプロセスについて、この処理を繰り返す。

【0090】図 23 は、図 3 のライブラリとしてユーザプロセス 12-1、12-2 にリンクされた 2 つのユーザレベルプロセススケジューラ 10-1、10-2 による動作を示す。プロセス識別子 A のユーザプロセス 12-1 にリンクされたユーザレベルプロセススケジューラ 10-1 は、図 22 の場合と同様、

kill システムコールによる実行、停止、
prioctl システムコールによる優先度変更、
prioctl システムコールによるクラス変更、
のいずれかを行う。

【0091】次に、ユーザプロセス 12-1 にリンクされたユーザレベルプロセススケジューラ 10-1 をプロセス管理表の CPU 使用時間だけ usleep システムコールによりスリープする。これにより、図 23 のようにユーザプロセス 12-1 を動作させる。次のプロセス識別子 B をもつユーザプロセス 12-2 についても、同様にして動作させる。

【0092】図 24 は、CPU の割当対象となるユーザプロセスが複数の場合の動作形態である。図 24 にあっては、ユーザレベルプロセススケジューラ 10 が 2 つのユーザプロセス 12-1、12-2 を動作する。まずユーザレベルプロセススケジューラ 10 は、

kill システムコールによる実行、停止、

prioctl システムコールによる優先度変更、
prioctl システムコールによるクラス変更、
のいずれかにより、プロセス識別子 A、B をもつ 2 つのユーザプロセス 12-1、12-2 を動作可能状態とする。

【0093】続いてユーザレベルプロセススケジューラ 10 は、usleep システムコールによりプロセス管理表から得られたユーザプロセス 12-1、12-2 の合計 CPU 時間で決まる一定時間スリープすることで、図示のように 2 つのユーザプロセス 12-1、12-2 を動作させる。この場合、ユーザレベルプロセススケジューラ 10 のスリープ期間にユーザプロセス 12-1、12-2 がどのように動作するかは、それぞれのユーザプロセス 12-1、12-2 の状態によって決まる。例えば、ユーザプロセス 12-1 が CPU を手放さなければユーザプロセス 12-2 は動作しないし、ユーザプロセス 12-1 が直ちに CPU を手放せばユーザプロセス 12-2 が動作するようになる。ここでユーザプロセス 12-1、12-2 が実行を停止する原因としては、I/O 待ち、sleep システムコールの発行等がある。

【0094】図 25 は、図 24 の複数のユーザプロセスを複数の CPU からなるマルチプロセッサ上で動作させる場合の形態である。この場合には、図 25 (A)

(B) のように、ユーザプロセス 12-1、12-2 のそれぞれに CPU 48-1、48-2 を割り当てることのできるため、異なる CPU 48-1、48-2 上で同時に 2 つのユーザプロセス 12-1、12-2 を動作させることができる。図 26 は、予め定めた一定時間 T で複数のユーザプロセスの動作を保証する実施形態である。

ユーザレベルプロセススケジューラ 10 は、

kill システムコールによる実行と停止、
prioctl システムコールによる優先度の変更、

prioctl システムコールによるクラス変更、
のいずれかにより、プロセス識別子 A のユーザプロセス 12-1 を一定時間動作させた後、同様にしてプロセス識別子 B のユーザプロセス 12-2 を一定時間動作させる。

【0095】図 27 は、複数のユーザプロセスを動作可能状態として一定周期で繰り返し動作させる形態の説明図である。この実施形態にあっては、2 つのユーザプロセス 12-1、12-2 をスケジューリング対象としており、一定の周期 T についてユーザレベルプロセススケジューラ 10 は、まずプロセス識別子 A のユーザプロセス 12-1 を、

kill システムコールによる実行、停止、
prioctl システムコールによる優先度変更、
prioctl システムコールによるクラス変更、
のいずれかで動作可能状態とした後、一定時間動作さ

せ、次に、同様にしてユーザプロセス 12-2 を動作さ

せ、これを周期的に繰り返す。この図 25 の周期 T による複数のユーザプロセスの動作の繰返しは、各ユーザプロセスの動作周期が同じ場合に適用できる。

【0096】図 28 は、複数のユーザプロセスの動作周期が異なる場合の動作形態である。まずユーザプロセス 12-1 は、図 28 (A) のように、周期 T で動作を繰り返している。これに対しユーザプロセス 12-2 は、図 28 (B) のように、2 倍の周期に T で動作を繰り返している。このように 2 つのユーザプロセス 12-1、12-2 の周期が異なる場合には、図 29 のように、複数の動作周期の最小公倍数である周期 2 T をスケジューリング周期とする。

【0097】即ち、周期 T のユーザプロセス 12-1 を動作した後、周期 2 T のユーザプロセス 12-2 を動作し、更に周期 T のユーザプロセス 12-1 を動作させ、残りを空き周期とし、これを繰り返す。ここで一定周期 T で複数のユーザプロセスを動作させる図 27 の実施形態にあっては、1 周期 T 内の CPU 時間を 100 % 使用しており、このような動作環境に周期 T 内で例えば 20 % の CPU 時間を要求する別のユーザプロセスをスケジューリング対象に加入しようとしても、この要求はユーザレベルプロセススケジューラ 10 により拒否されてしまう。

【0098】これに対し、もしユーザプロセス 12-1、12-2 の周期 T 内での CPU 使用率の合計が 80 % 以下に制限されていれば、他のユーザプロセスの CPU 時間 20 % の要求を受理してユーザレベルプロセススケジューラのスケジュール対象に加入させることができる。そこで、図 27 の一定周期 T の動作形態についてクオリティ・オブ・セレクト制御の 1 つであるアドミッションコントロールを採用し、1 周期 T 内におけるユーザプロセス 12-1、12-2 の CPU 使用時間を 80 % 以下に制限する。これによって、新たな 20 % 以内の CPU 使用率をもった他のユーザプロセスの加入要求を効率良く受け入れることができる。

【0099】図 30 は、一定期間内で CPU を分割してユーザプロセスに割り当てる動作形態である。即ち図 26 または図 27 にあっては、一定時間 T または一定周期内に連続してユーザプロセスの CPU 割当てを行っているが、図 30 にあっては、一定期間の CPU 時間を複数に分割し、各分割時間についてユーザプロセス 12-1 とユーザプロセス 12-2 を割り当てている。

【0100】図 31 は、タイムシェアリングクラスに存在するスケジューリング対象となっているユーザプロセスに対し、強制的に一定の CPU 時間割当てを行う動作形態の説明図である。図 31 において、ユーザプロセス 12-1、12-2 はリアルタイムクラス 28 に存在し、ユーザプロセス 12-3、12-4 はタイムシェアリングクラス 30 に存在している。この場合、ユーザレベルプロセススケジューラ 10 は、

kill システムコールによる実行停止、

priocntl システムコールによる優先度変更、

priocntl システムコールによるクラス変更、

のいずれかで、リアルタイムクラス 28 にあるユーザプロセス 12-1、12-2 を順次一定時間動作させた後、ユーザプロセス 12-1、12-2 を実行停止状態とし、usleep システムコールにより、一定時間 CPU を手放す。

【0101】これにより CPU はタイムシェアリングクラス 30 に割り当てられ、ユーザプロセス 12-3、12-4 を動作させる。この場合のユーザプロセス 12-3、12-4 のスケジューリングは、オペレーティングシステム Solaris 2.4 がもつタイムシェアリングスケジューラが行うことになる。図 32 は、本発明のユーザレベルプロセススケジューラのスケジュール空間となるリアルタイムクラスとグローバルな優先度の対応関係を示したマッピングの説明図である。

【0102】本発明のユーザレベルプロセススケジューラをサポートするオペレーティングシステム Solaris 2.4 では、タイムシェアリングクラス 30 については、図 32 (C) のように、タイムシェアリングクラス優先度 0 ~ 59 をグローバル優先度 0 ~ 59 にマッピングさせ、設定しており、リアルタイムクラス 28 については、図 32 (A) のように、リアルタイムクラス優先度 0 ~ 59 をグローバル優先度 100 ~ 159 にマッピングさせ、設定している。

【0103】この図 32 (A) の優先度のマッピングは、オペレーティングシステムが起動するときにリンクする RT-DPTBL というファイルに転記されている。本発明にあっては、図 32 (A) のマッピングファイルを、図 32 (B) に示すようにリアルタイムクラス 28 の最低優先度をタイムシェアリングクラス 30 の最低優先度 0 と等しくする。

【0104】図 32 (B) のマッピングによれば、図 33 の動作形態が得られる。図 7 のプロセス優先度制御部 26 に対する指示でユーザプロセス 12-2 を優先度 0 に設定することで、一定時間をタイムシェアリングプロセス用時間 T_{ts} に割り当てる。この場合、タイムシェアリングプロセス用時間 T_{ts} の間、実際には実行可能なタイムシェアリングプロセスが存在しなかったとすると、図 32 (B) の最低優先度 0 をもつリアルタイムクラス 28 のユーザプロセス 12-2 をタイムシェアリングクラスのユーザプロセスの代わりに動作させることができる。

【0105】もちろん、このときユーザプロセス 12-1 は実行停止状態となっていることから、ユーザプロセス 12-2 の動作中に動作することはない。これによって、一定時間タイムシェアリングクラスのユーザプロセスに動作時間を割り当てても効率良くリアルタイムクラスのユーザプロセスを動作させることができる。

5. ユーザプロセスのブロック検出

図34は、ユーザレベルプロセススケジューラ10によりユーザプロセス12-1に時間T1からT4までCPU時間Tを割り当てた状態でブロックが発生した場合である。即ち、時間T1でユーザプロセス12-1がCPU時間の割当てを受けて動作を開始したが、時刻T2でI/O待ち等でブロック54が発生したとする。

【0106】このブロック54は適宜のブロック検出手段により検出されて、ブロック検出通知56がユーザレベルプロセススケジューラ10に通知される。ユーザレベルプロセススケジューラ10はブロック検出56の通知を受けると、残り時間T3~T4をタイムシェアリングクラスのユーザプロセスに割り当てる。図35は、図34におけるブロック検出のためのプロセスの動作形態である。

【0107】図35において、優先度159のユーザレベルプロセススケジューラ10のスケジュール対象となるユーザプロセス12-1と同じ優先度158にブロック検出プロセス60を設けている。本発明をサポートするオペレーティングシステムSolaris2.4のリアルタイムクラスでは、同一優先度に複数の実行可能なプロセスが存在する場合、CPU割当ての最小単位であるタイムカンタムごとにラウンドロビンで実行可能プロセスを切り替える。またタイムカンタムは、プロセスごとにpriorityシステムコールで設定することができる。

【0108】図35にあっては、ユーザレベルプロセススケジューラ10がユーザプロセス12-1とブロック検出プロセス60を次の手順で設定し、図34のようにユーザプロセス12-1がブロックしたことをブロック検出プロセス60で検出し、オペレーティングシステムのシグナル機能を使用してユーザレベルプロセススケジューラ10に通知することになる。

【0109】即ち、図35のユーザレベルプロセススケジューラ10によるブロック検出プロセス60の設定は次のようになる。

ユーザプロセス12-1を優先度158に設定する。

ユーザプロセス12-1のタイムカンタムを、ユーザプロセス12-1を与えるCPU時間と等しい時間またはそれ以上の値をもつ時間に設定する。

【0110】 ユーザプロセス12-1を実行可能状態とする。

ブロック検出プロセス60を優先度158に設定する。

ブロック検出プロセス60のタイムカンタムを適当な時間に設定する。

ブロック検出プロセス60を実行可能状態とする。

ユーザプロセス12-1に与えたCPU時間だけスリープする。

【0111】図37はブロック検出プロセスの他の動作

形態であり、ユーザレベルプロセススケジューラ10のスケジュール対象となるユーザプロセス12-1に対し更に下位の優先度158にブロック検出プロセス60を設定している。即ち、ユーザプロセス12-1がブロックしたことをブロック検出プロセス60で検出し、オペレーティングシステムのシグナル機能を使用してユーザレベルプロセススケジューラ10に通知する。この場合のユーザプロセス12-1及びブロック検出プロセス60の設定は次のようになる。

10 【0112】 ユーザプロセス12-1を優先度158に設定する。

ユーザプロセス12-1を実行可能状態とする。

ブロック検出プロセス60を優先度157に設定する。

ブロック検出プロセス60を実行可能状態とする。

ユーザプロセス12-1を動作させたい時間スリープする。

20 【0113】この図37のブロック検出プロセス60の設定の場合の動作も、図36と同じになる。図38は、ユーザプロセスに割り当てた一定のCPU時間内にブロック検出が行われた後、実行可能状態となった場合の動作形態の説明図である。まずユーザレベルプロセススケジューラ10は、ユーザプロセス12-1に時間T1からT6までの時間をCPU時間として割り当てている。しかしながら、ユーザプロセス12-1が時間T2でI/O待ち等でブロックし、ブロック検出68を受けてユーザレベルプロセススケジューラ10は残り時間T3からT6をタイムシェアリングクラスのプロセス120に割り当てる。

30 【0114】ところが、その途中の時間T4でユーザプロセス12-1が実行可能になったとすると、適宜の検出手段により実行可能検出72がユーザレベルプロセススケジューラ10に通知され、この通知を受けて時間T5からT6を再度ユーザプロセス12-1に割り当てる。図38における一度ブロックしたユーザプロセス12-1の実行可能状態の検出は、オペレーティングシステムからの通知により可能となる。

【0115】図39は、ユーザプロセスの生成と終了を検出した場合の動作形態の説明図である。まず図39

40 (A)は、ユーザプロセス12-1のみを動作しており、CPU時間の一部をタイムシェアリングプロセス120用に割り当てていた場合である。この状態で新たなユーザプロセス12-2が生成されると、図39(B)のように、ユーザプロセス12-2の生成を検出し、タイムシェアリングプロセスに割り当てていたCPU時間をユーザプロセス12-2に割り当てる。

【0116】一方、ユーザレベルプロセススケジューラ10は、図39(B)のように、ユーザプロセス12-1、12-2にCPU時間を割り当てていたが、ユーザプロセス12-2が終了したことを検出した場合、この

ユーザプロセス 12-2 の CPU 割当時間をタイムシェアリングプロセス 120 に割り当て、図 39 (A) の動作形態とする。

【0117】図 39 のようなプロセスの生成、終了については、プロセス生成を行うシステムコール fork とプロセスの終了を行う exit システムコールで行われるが、C 言語で書かれた図 40 (A) のアプリケーション 32 の場合、ライブラリ 34 を通してオペレーティングシステム 14 に通知される。また本発明のユーザレベルプロセススケジューラ 10 が存在する場合、図 40 (B) のように、アプリケーションプログラム 32 よりプロセス生成のシステムコール fork またはプロセスの終了のシステムコール exit を発行した場合、ユーザレベルプロセススケジューラ 10 に対しメッセージ通信機能をもつ新ライブラリ 340 を設ける。

【0118】このため新ライブラリ 340 は、アプリケーションプログラム 32 よりシステムコール fork または exit を受けると、プロセス間通信機能の 1 つであるメッセージを使用してユーザレベルプロセススケジューラ 10 に生成または終了を通知することができる。この新ライブラリ 340 については、コンパイル時にスタティックリンクされたものは再リンクを行い、実行時にダイナミックリンクされるものに対してはダイナミックリンクライブラリを置き替える。これにより、アプリケーションプログラム 32 を書き替える必要はない。

【0119】図 40 (B) の形態でユーザプロセスの生成、終了を検出した場合の動作は、図 39 (A) (B) と同じになる。もちろん、ユーザプロセスの生成と終了はオペレーティングシステム 14 から直接ユーザレベルプロセススケジューラ 10 に通知して行うようにしてもよい。尚、上記の実施例は、固定優先度スケジューリングをリアルタイムクラスとして提供するサンマイクロシステム社のオペレーティングシステム Solaris 2.4 を例にとるものであったが、他の固定優先度スケジューリング方式を提供する適宜のオペレーティングシステムについてそのまま適用できる。

【0120】また本発明のユーザレベルプロセススケジューラがリンクする固定優先度のスケジューラに対する他のクラスのスケジューラとして、タイムシェアリングスケジューラを例にとっているが、これ以外の優先度可変のスケジューラであってもよいことはもちろんである。

【0121】

【発明の効果】以上説明してきたように本発明によれば、マルチメディア処理で必要とされる一定割合の CPU 利用を保証するスケジューラをユーザのニーズに合わせて様々な形態で商用のオペレーティングシステム上に実装することができる。またユーザレベルで実現されているにもかかわらず、振る舞いの信用できないプロセスをスケジューリング対象とすることができるなど、オペ

レーティングシステム内でスケジューラを実現すると同等のスケジューラをユーザレベルで実装することができる。

【図面の簡単な説明】

【図 1】本発明の原理説明図

【図 2】ユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図 3】並列型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

10 【図 4】階層型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図 5】図 3 を対象とした階層型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図

【図 6】同一優先度に 2 つのユーザレベルプロセススケジューラを設けた動作環境の説明図

【図 7】図 2 のユーザプロセススケジューラの詳細説明図

【図 8】図 7 のプロセス管理表の説明図

【図 9】ユーザプロセスの実行、停止による動作説明図

20 【図 10】図 7 のプロセス実行制御部による実行と停止の処理のフローチャート

【図 11】ユーザプロセスの優先度変更による動作説明図

【図 12】図 7 のプロセス優先度制御部による実行と停止の処理のフローチャート

【図 13】ユーザプロセスのクラス変更による動作説明図

【図 14】図 7 のクラス変更制御部による実行と停止の処理のフローチャート

30 【図 15】図 7 の実行停止を用いたユーザプロセスの加入脱会処理のフローチャート

【図 16】図 7 の優先度変更を用いたユーザプロセスの加入脱会処理のフローチャート

【図 17】図 7 のクラス変更を用いたユーザプロセスの加入脱会処理のフローチャート

【図 18】ライブラリを用いたスケジュール変更要求の説明図

【図 19】ライブラリのスケジュール仕様に基づくスケジュール変更要求の説明図

40 【図 20】図 19 のスケジュール仕様の説明図

【図 21】マルチスレッド構成をもつユーザプロセスの説明図

【図 22】図 2 の動作環境で一定時間に 1 つのユーザプロセスのみを動作させる説明図

【図 23】図 3 の動作環境で一定時間に 1 つのユーザプロセスのみを動作させる説明図

【図 24】CPU 割当て対象のユーザプロセスを複数動作させた説明図

50 【図 25】複数の CPU を割り当てて動作させた場合の説明図

【図 2 6】一定時間内に複数のユーザプロセスを動作させる説明図

【図 2 7】一定周期で複数のユーザプロセスを動作させる説明図

【図 2 8】動作周期の異なるユーザプロセスの説明図

【図 2 9】図 2 8 のユーザプロセスを一定周期で動作させた説明図

【図 3 0】一定期間を複数分割して複数のユーザプロセスを動作させた説明図

【図 3 1】タイムシェアリングプロセスに CPU 時間を割り当てた説明図

【図 3 2】タイムシェアリングクラスとリアルタイムクラスの優先度のマッピングの説明図

【図 3 3】リアルタイムクラスの優先度 0 にマッピングしたユーザプロセスの動作の説明図

【図 3 4】ユーザプロセスのブロック検出時の動作の説明図

【図 3 5】ブロック検出プロセスの動作環境の説明図

【図 3 6】ブロック検出プロセスのブロック検出と動作の説明図

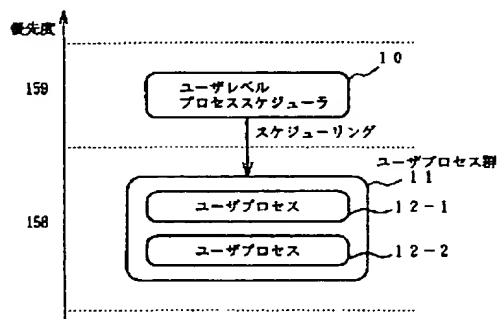
【図 3 7】ブロック検出プロセスの他の動作環境の説明図

【図 3 8】ブロック検出後に実行可能が検出された場合の説明図

【図 3 9】ユーザプロセスの生成と終了の説明図

【図 2】

ユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



【図 8】

図 7 のプロセス管理表の説明図

プロセス識別子	周期内の CPU 使用時間 (ms)
A	50
B	20
C	10
未使用	-
未使用	-

【図 4 0】アプリケーションプログラムのライブラリを使用したユーザプロセスの発生終了の検出通知の説明図

【符号の説明】

10, 10-1, 10-2, 10-11, 10-12, 1-21, 10-22: ユーザレベルプロセススケジューラ

12, 12-1, 12-2 ~ 12-4: ユーザプロセス

14: オペレーティングシステム (OS)

16: クラス変更指示部

18: プロセス管理表

20: プロセス実行指示部

22: クラス変更制御部

24: プロセス実行制御部

26: プロセス優先度制御部

28: リアルタイムクラス (固定優先度スケジューラクラス)

30: タイムシェアリングクラス (動的優先度スケジューラクラス)

32: アプリケーションプログラム

34, 38: ライブラリ

36: 親プロセス

42: マルチスレッドライブラリ

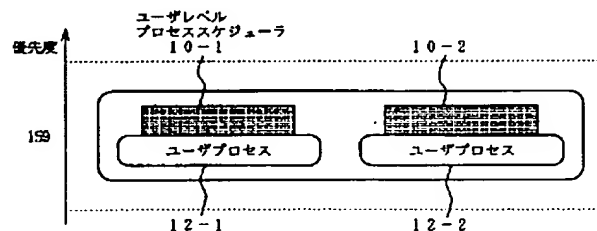
44-1 ~ 44-3: スレッド

60: ブロック検出プロセス

340: 新ライブラリ

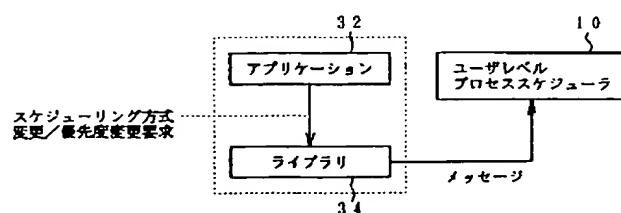
【図 3】

並列型のユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



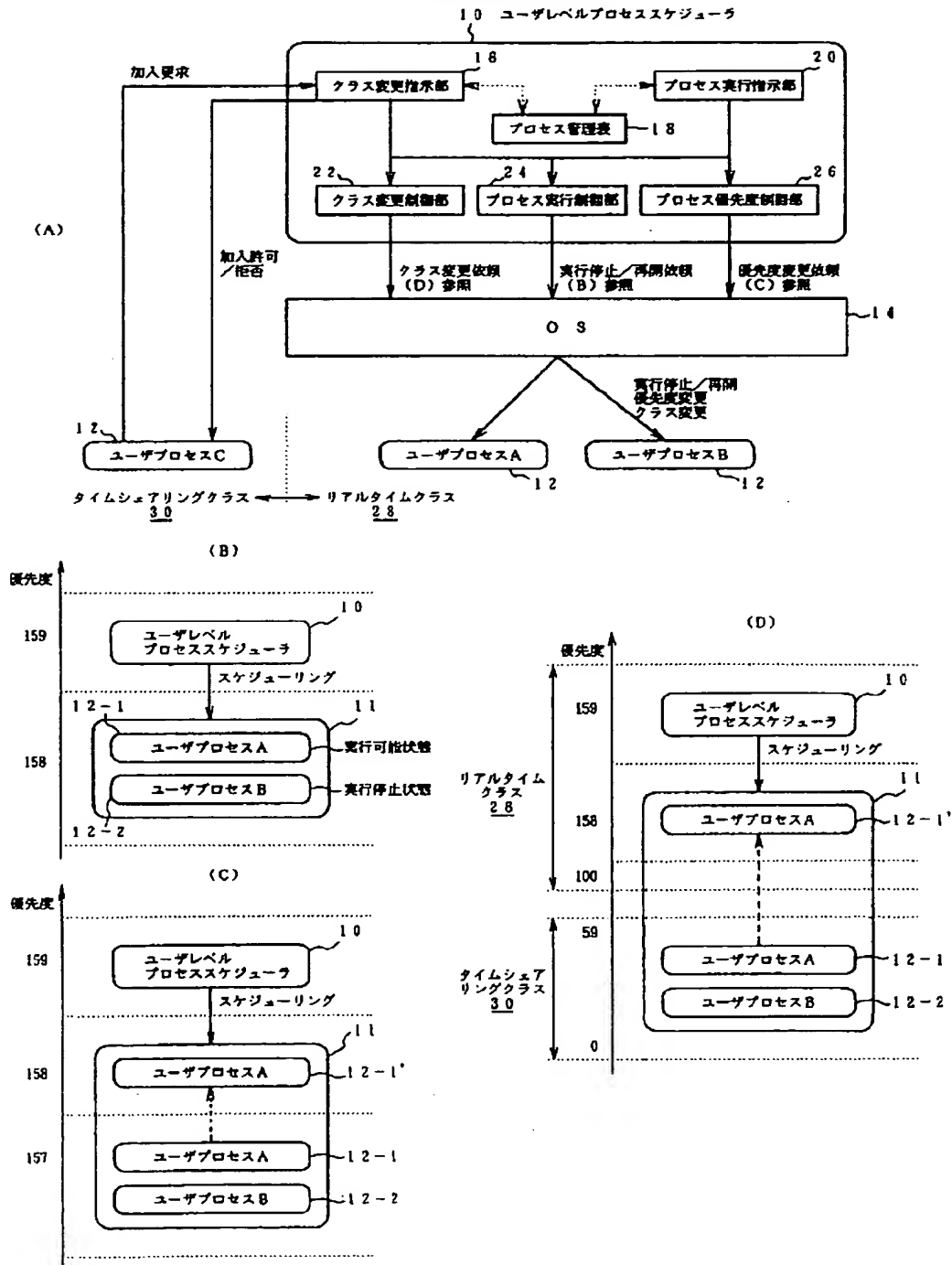
【図 18】

ライブラリを用いたスケジュール変更要求の説明図



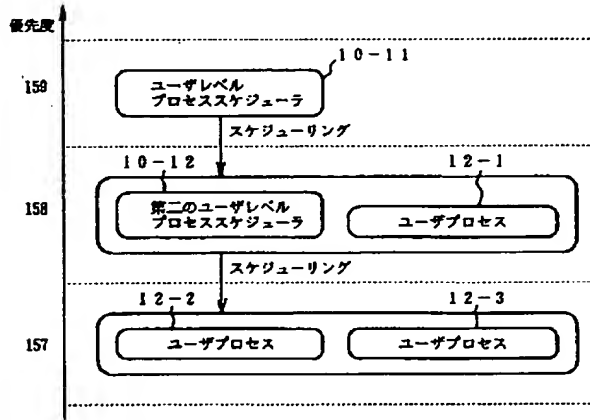
【図 1】

本発明の原理説明図



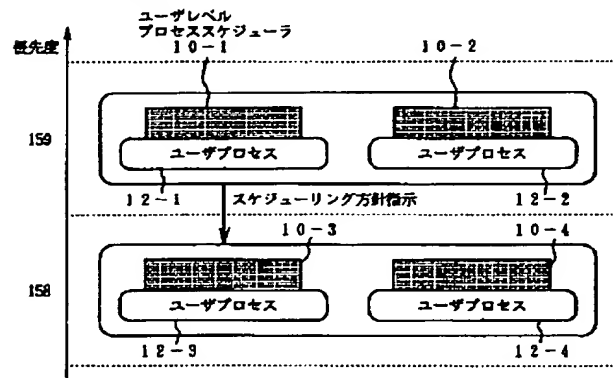
【図 4】

階層型ユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



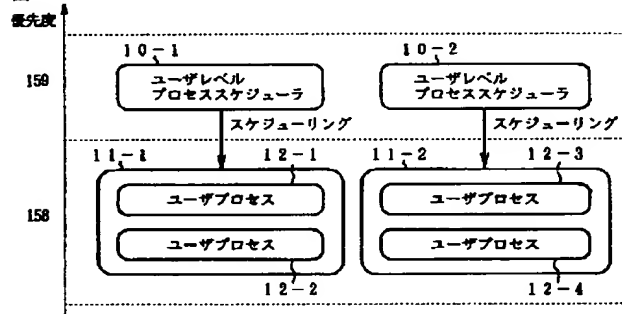
【図 5】

図3を対象とした階層型ユーザレベルプロセススケジューラとユーザプロセスの動作環境の説明図



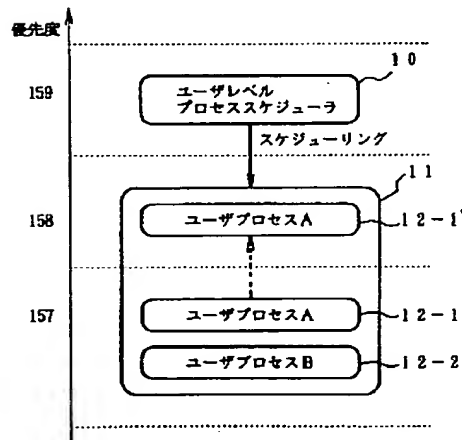
【図 6】

同一優先度で2つのユーザレベルプロセススケジューラを設けた動作環境の説明図

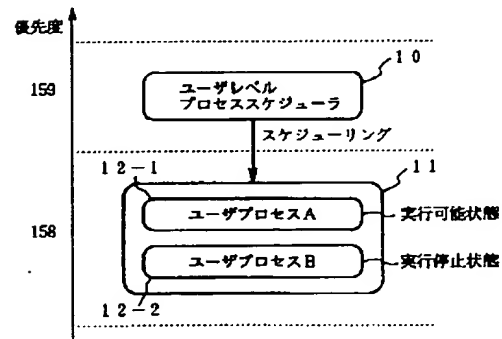


【図 11】

ユーザプロセスの優先度変更による動作説明図

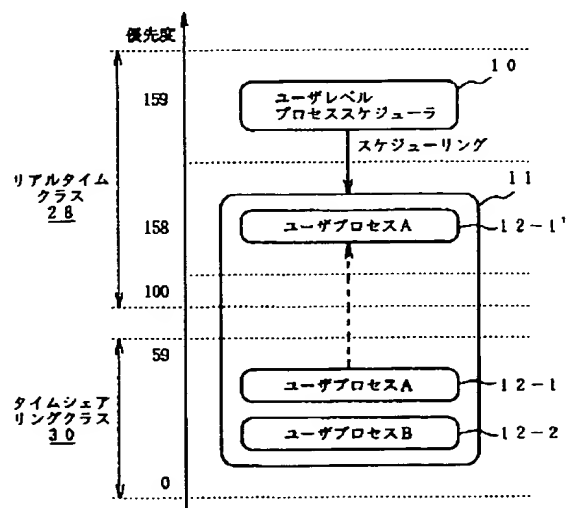


ユーザプロセスの実行、停止による動作説明図



【図 13】

ユーザプロセスのクラス変更による動作説明図



【図 7】

図2のユーザプロセススケジューラの詳細説明図

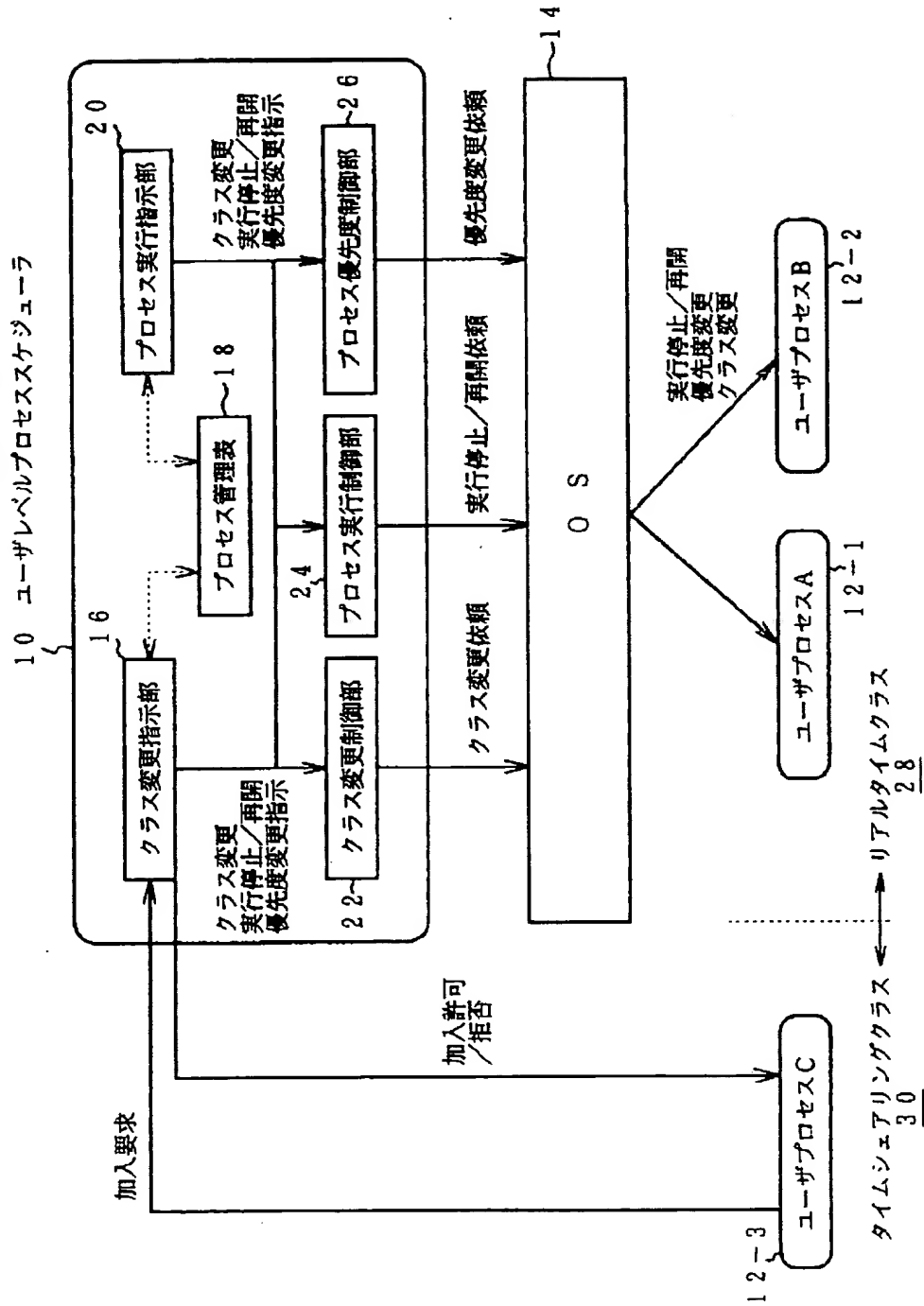
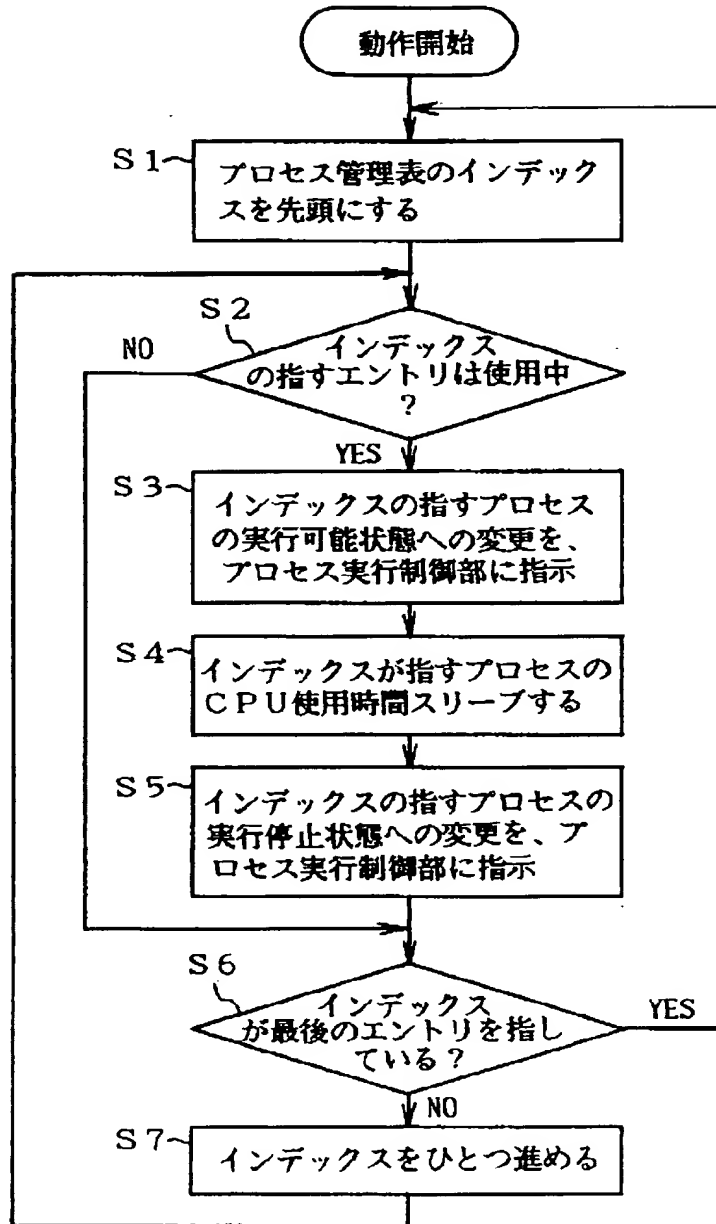


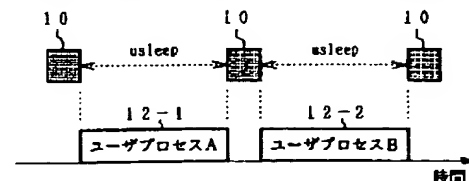
図7のプロセス実行制御部による実行と停止の処理のフローチャート



【図 2 2】

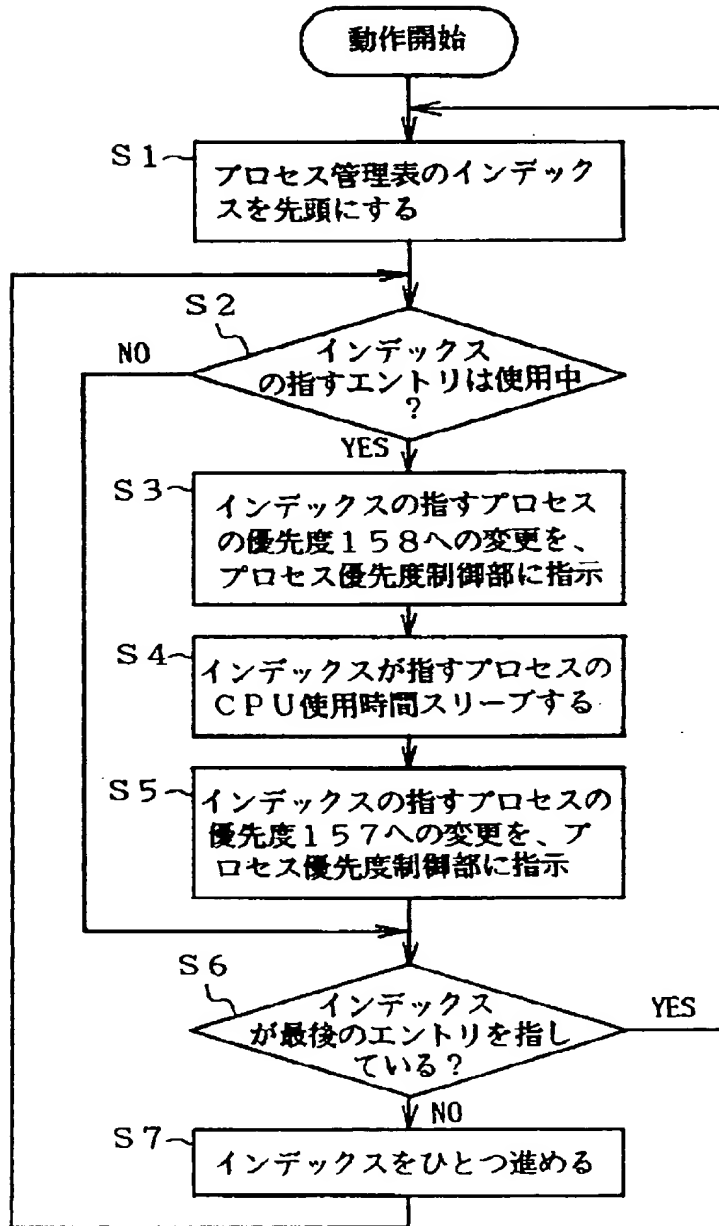
図2の動作環境で一定時間に1つのユーザプロセスのみを動作させる説明図

開始時刻	終了時刻	スケジューリング方式
00:00:00	00:00:30	タイムシェアリングクラス
00:00:30	00:10:00	リアルタイムクラス
00:10:00	00:10:30	タイムシェアリングクラス



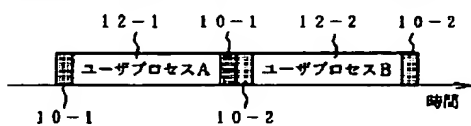
【図 12】

図 7 のプロセス優先度制御部による実行と停止の処理のフローチャート



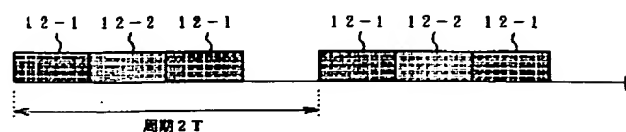
【図 23】

図 3 の動作環境で一定時間に 1 つのユーザプロセスのみを動作させる説明図



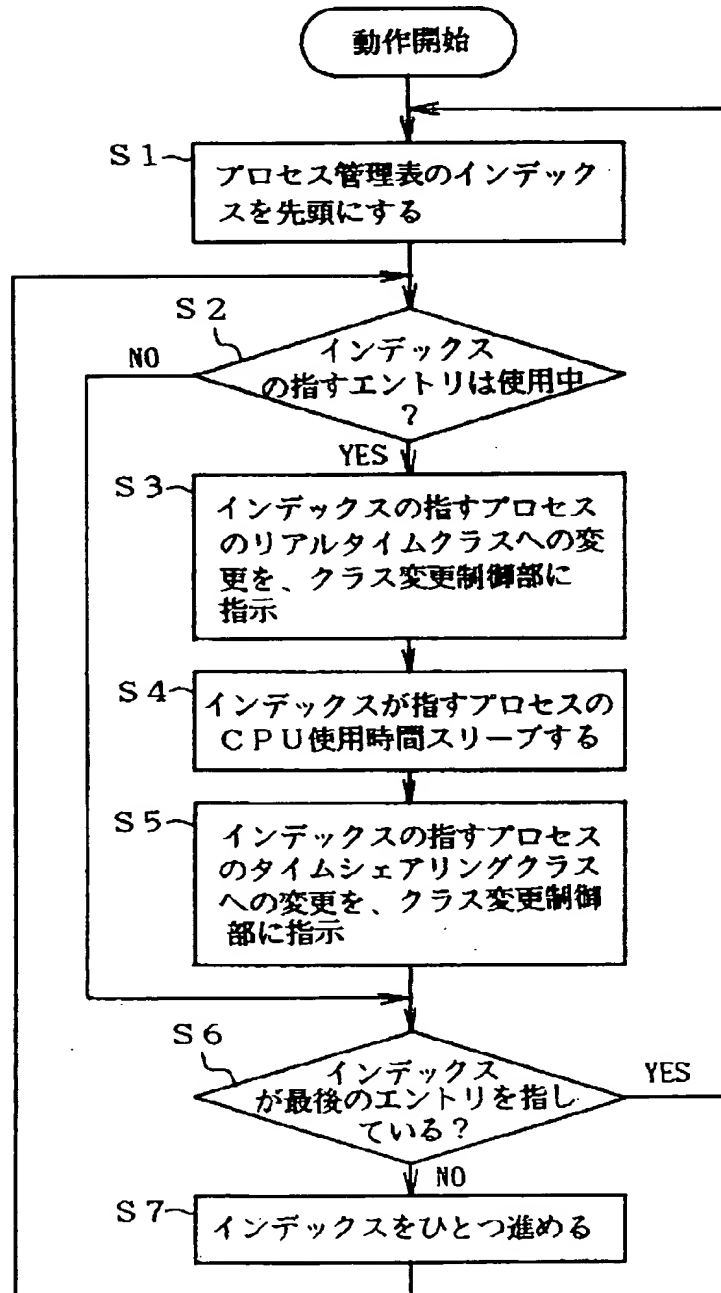
【図 29】

図 28 のユーザプロセスを一定周期で動作させた説明図



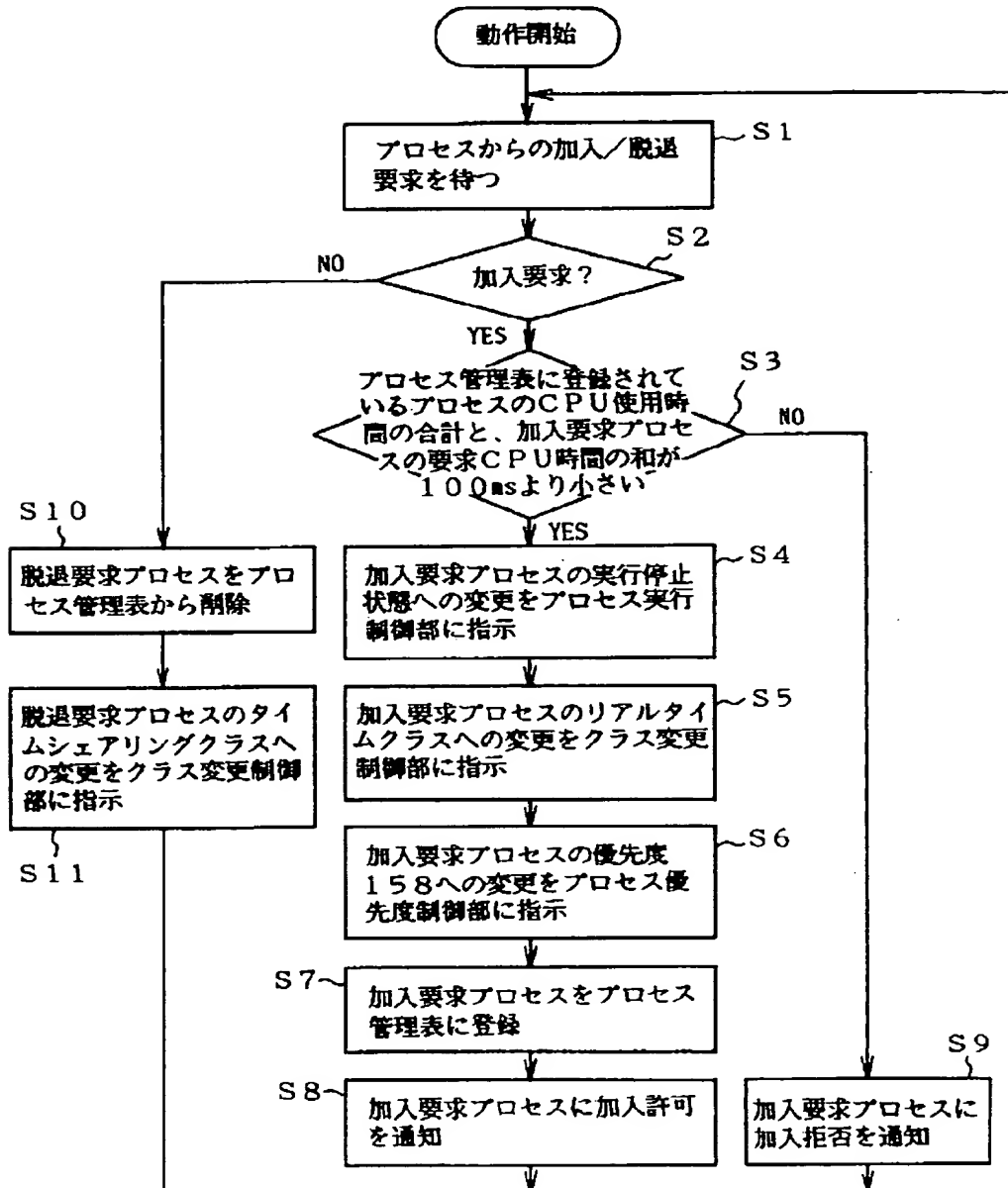
【図 1 4】

図 7 のクラス変更制御部による実行と停止の処理のフローチャート



【図 15】

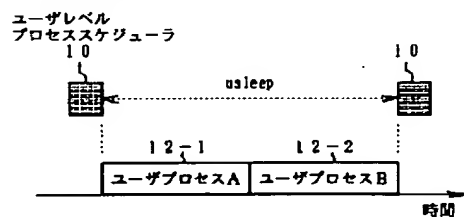
図 7 の実行停止を用いたユーザプロセスの加入脱会処理のフローチャート



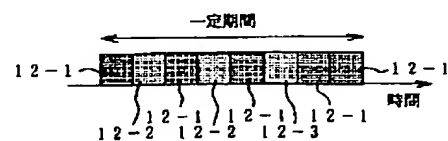
【図 24】

【図 30】

CPU割当て対象のユーザプロセスを複数動作させた説明図

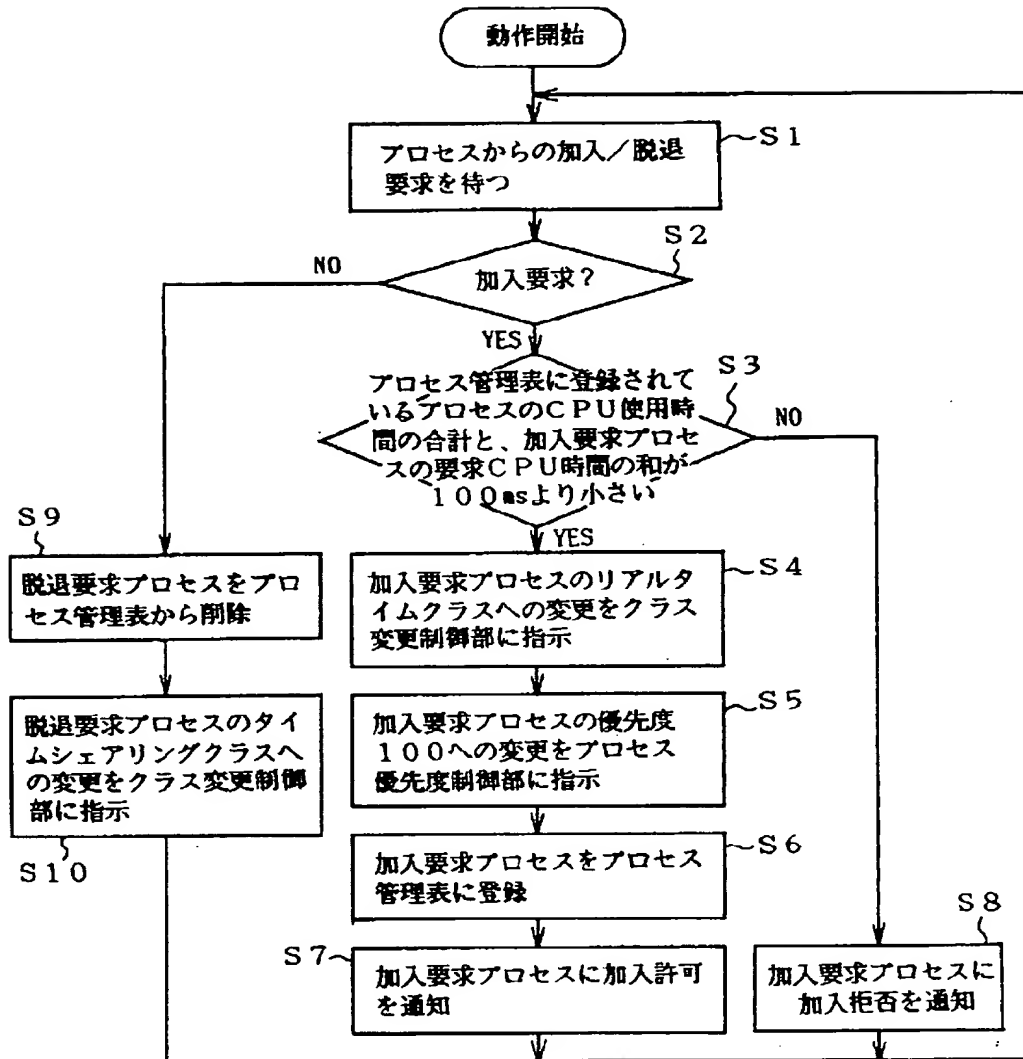


一定期間を複数分割して複数のユーザプロセスを動作させた説明図



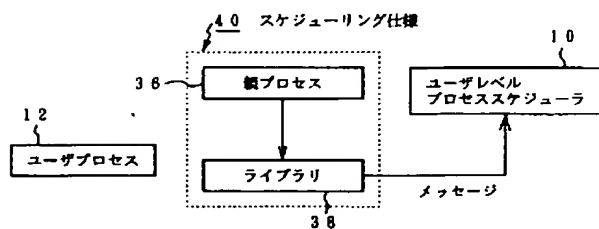
【図 16】

図 7 の優先度変更を用いたユーザプロセスの加入脱会処理のフローチャート



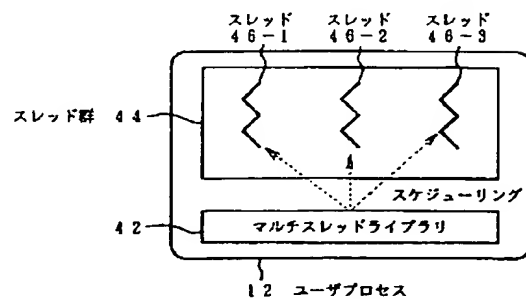
【図 19】

ライブラリのスケジュール仕様にに基づくスケジュール変更要求の説明図



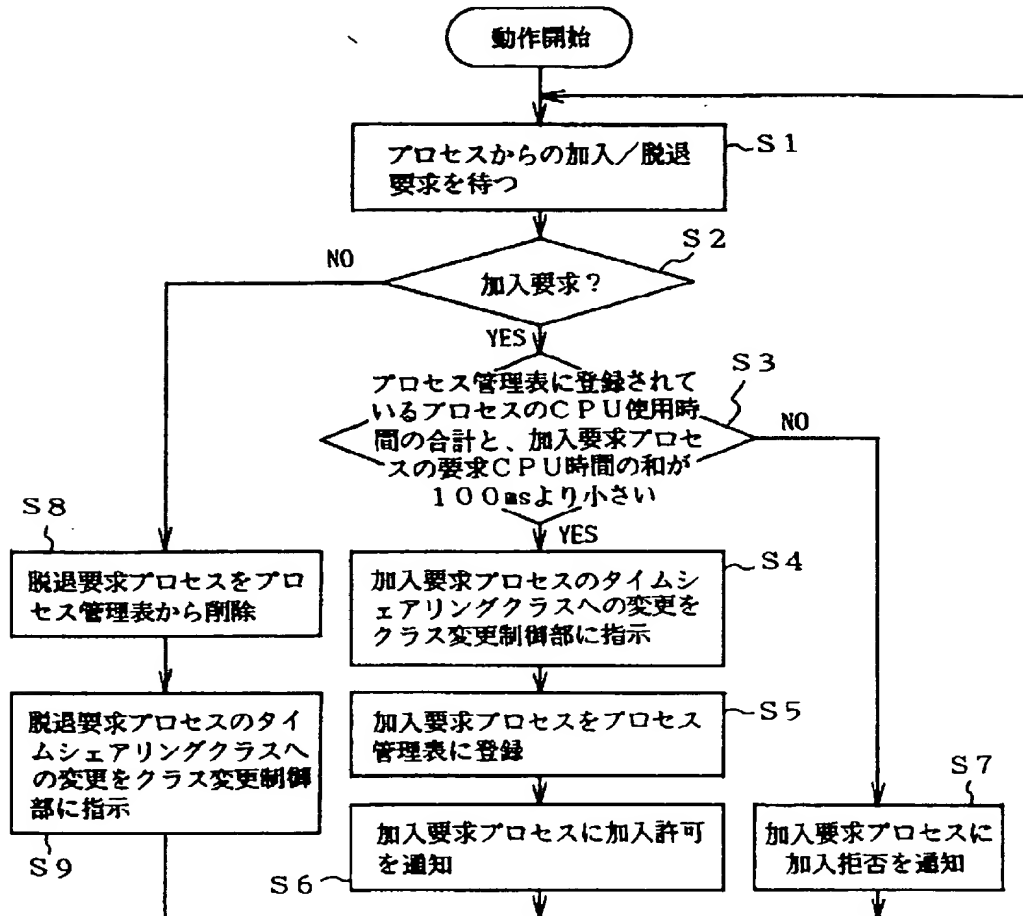
【図 21】

マルチスレッド構成をもつユーザプロセスの説明図



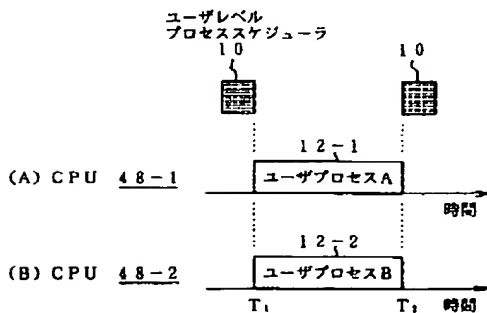
【図 17】

図 7 のクラス変更を用いたユーザプロセスの加入脱会処理のフローチャート



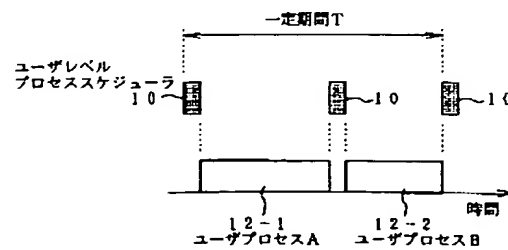
【図 25】

複数のCPUを割り当てて動作させた場合の説明図



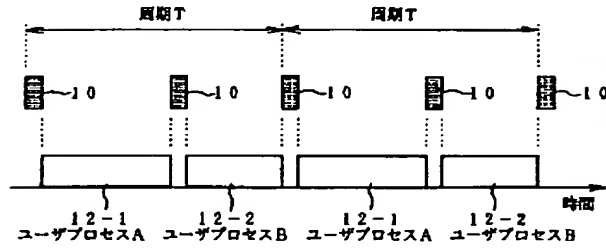
【図 26】

一定時間内に複数のユーザプロセスを動作させる説明図



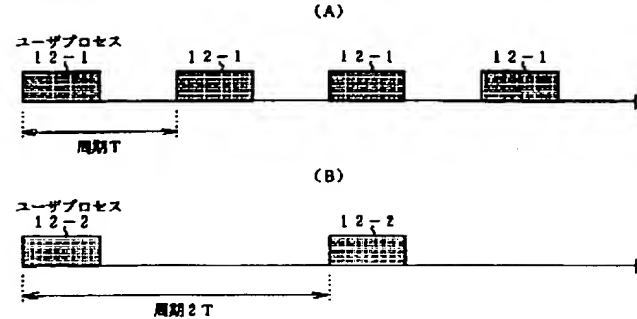
【図 27】

一定周期で複数のユーザプロセスを動作させる説明図



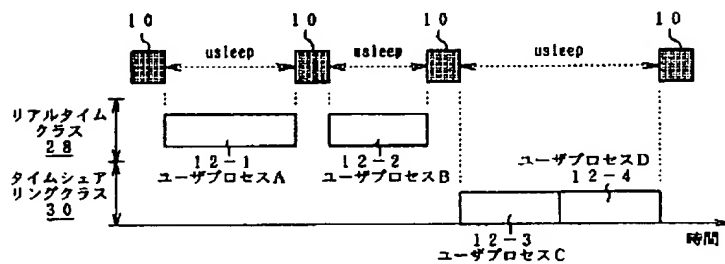
【図 28】

動作周期の異なるユーザプロセスの説明図



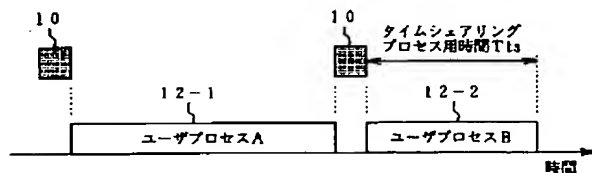
【図 31】

タイムシェアリングプロセスにCPU時間を割り当てた説明図



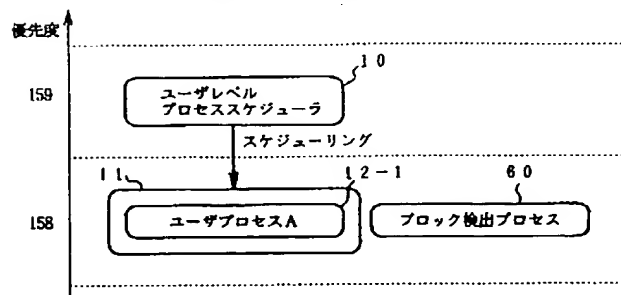
【図 33】

リアルタイムクラスの優先度0にマッピングしたユーザプロセスの動作の説明図



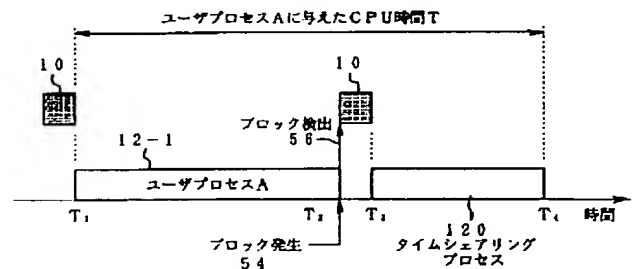
【図 35】

ブロック検出プロセスの動作環境の説明図



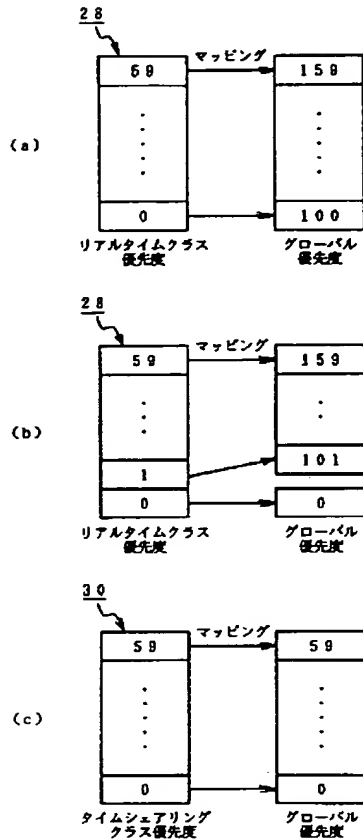
【図 34】

ユーザプロセスのブロック検出時の動作の説明図



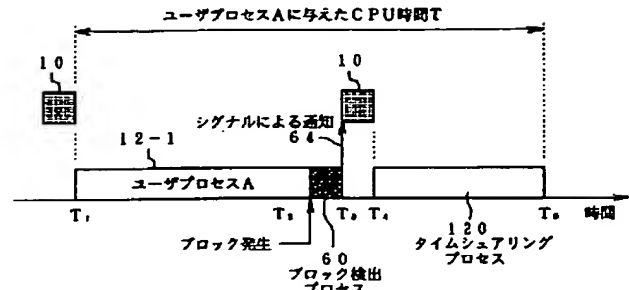
【図 3 2】

タイムシェアリングクラスとリアルタイムクラスの優先度のマッピングの説明図



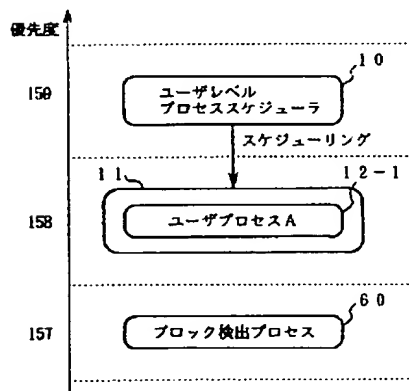
【図 3 6】

ブロック検出プロセスのブロック検出と動作の説明図



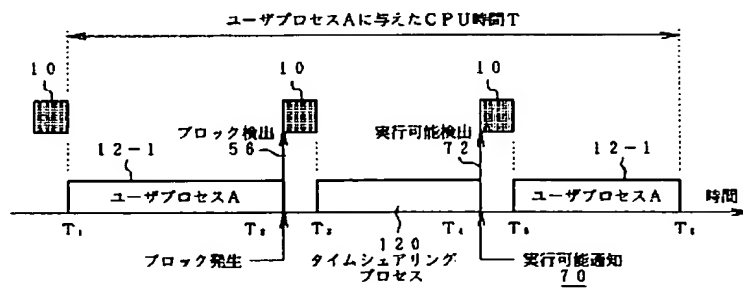
【図 3 7】

ブロック検出プロセスの他の動作環境の説明図



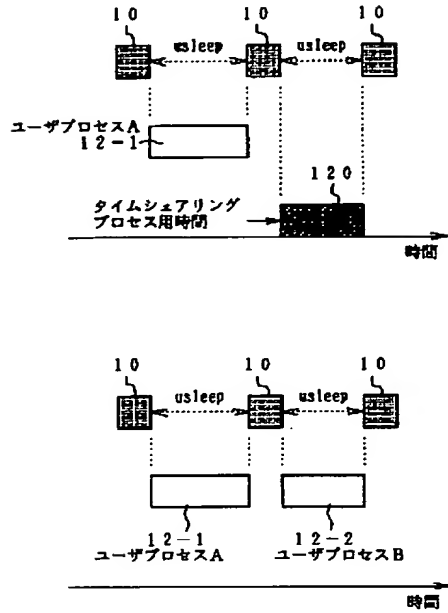
【図 3 8】

ブロック検出後に実行可能が検出された場合の説明図



【図 39】

ユーザプロセスの生成と終了の説明図



【図 40】

アプリケーションプログラムのライブラリを使用したユーザプロセスの発生終了の検出通知の説明図

